Waste Reduction and Design Problem in a Veranda Company: The Case of Veranco

Barcelona, June 2006
The author and the promotors give the permission to use this thesis for consultation and to copy parts of it for personal use. Every other use is subject to the copyright laws; more specifically the source must be extensively specified when using results from this thesis.
Abstract

Veranco is a company producing customized verandas, ‘prêt-à poser’, in aluminium or wood. This thesis describes two problems the company is facing nowadays. The first problem is a cutting stock problem with low demands. Sawing the seven-meter stock profiles to the desired length, results in a huge amount of waste material. The company estimates the waste at 25%.

For the waste reduction problem we suggest the following procedure: making the roof profile BOM; dividing the long list of roof profiles in different groups according to their colour; making a further subdivision for the different profile types; and sawing the packages of the same profile and colour in an optimized way.

Each package of the same profile and colour is a cutting stock problem. We present two algorithms: a low demand column generation algorithm and an exact algorithm. The first algorithm is an improvement of Gilmore and Gomory’s algorithm for low demands. The exact algorithm gives an exact solution without changing the Branch-and-Bound code. A smart fraction of all patterns is generated. A reduction of approximately 75% is reached. The two algorithms are programmed in Visual Basic, and the cutting stock problem was solved utilizing lp_solve, an open-source mixed integer linear programming solver based on the revised simplex method and the Branch-and-Bound method for integer variables.

With the optimizing program, we get a waste of 15.6%, a reduction of 10.2%. Another advantage is that fewer profiles have to be laid back in the stock. A reduction of 42.9% is obtained, which results in a better exploit of the warehouse surface. Moreover, time can be gained when telescoping roof profiles belonging together.

The second problem we investigate is a design problem occurring when cutting the angles of a rising gutter. A combination of a vertical and horizontal cutting angle has to be sawed. These angles are not just the half of the vertical and horizontal angles of the rising gutter, because the cuts are not independent of each other. The vertical cutting angle, the horizontal cutting angle and the torsion angle are calculated in function of the given vertical and horizontal angle of the rising gutter. This results in a profitable gain of production time for the company.
Acknowledgements

I would like to gratefully acknowledge the enthusiastic supervision of Prof. Manel Mateo Doll for the stimulating suggestions, critique, and sound advice. His constructive comments on the text have greatly improved this work. I am also indebted to Prof. El-Houssaine Aghezzaf for many insightful conversations during the development of the ideas in this thesis, providing resources and penetrating criticism, and offering directions.

Furthermore, I have to thank all the people from Veranco. Special thanks go to Rudi Wuytack, technical director of the company and Stefan Bosman, responsible for the R&D department.

I am very grateful for my years at the University of Ghent. It has been a great pleasure to study last semester at the Technical University of Catalonia (UPC), made possible by the Erasmus programme. Thanks go to Prof. Hendrik Van Landeghem, who made this bilateral agreement.

Thanks are also due to all my friends, on both sides of the ocean, for their warm support and encouragement, especially Nathaly Castro and Lennert Van Guyt. My flatmates Ohad Loffler Van Stokkum, Merlin Schwertner Precht and Juan Sanchez Vidal enabled me to complete this thesis and have a wonderful time along the way.

On a different note, I would like to thank: the banana producers of Ecuador for keeping me healthy, the L.A. fitness centre of Barcelona for keeping me fit, and the music of Columbian-born Shakira for keeping me sane.

Finally, I have to say 'thank-you' to my family, particularly my Mum and Dad, for their understanding, endless patience and encouragement when it was most required. This work is dedicated to them.

Peace and Solidarity,
Nicolas.
Barcelona, 15th June 2006
Contents

Introduction ............................................................................................................1

Chapter 1  Product Overview .............................................................................3
  1.1 The company .................................................................................................3
  1.2 What’s a veranda? ..........................................................................................5
  1.3 Veranda vocabulary .........................................................................................9
  1.4 Product families ............................................................................................12
    1.4.1 Roof profile types ......................................................................................15
      1.4.1.1 Ariane ....................................................................................................15
      1.4.1.2 Confort .................................................................................................15
      1.4.1.3 Prestige+ ..............................................................................................15
    1.4.2 Standard shapes .........................................................................................16
      1.4.2.1 Agate .....................................................................................................16
      1.4.2.2 Topaze ................................................................................................16
      1.4.2.3 Grenat ..................................................................................................17
      1.4.2.4 Emeraude ............................................................................................18
      1.4.2.5 Diamant ...............................................................................................19
      1.4.2.6 Victorienne .........................................................................................20
      1.4.2.7 Rubis ...................................................................................................21
    1.4.3 Roofing materials ......................................................................................22
      1.4.3.1 Polycarbonate ......................................................................................22
      1.4.3.2 Sandwich panel ...................................................................................22
      1.4.3.3 Double glazing ....................................................................................22

Chapter 2  Data Extraction and P-Q Analysis ..................................................23
  2.1 Data extraction ..............................................................................................23
    2.1.1 Roof profile type, thermal break and standard shape ............................23
    2.1.2 Roof material ..........................................................................................26
    2.1.3 Colour ......................................................................................................29
  2.2 P-Q analysis ..................................................................................................30
    2.2.1 Roof profile types ....................................................................................31
    2.2.2 Thermal break ..........................................................................................32
    2.2.3 Standard shapes .......................................................................................33
    2.2.4 Roofing materials ....................................................................................34
      2.2.4.1 Polycarbonate 25 mm ........................................................................35
      2.2.4.2 Sandwich panel ..................................................................................36
    2.2.5 Colours .....................................................................................................37
  2.3 Elaborate example of a veranda project and BOM ....................................38

Chapter 3  Veranda Production Process ............................................................49
  3.1 General overview ..........................................................................................49
  3.2 Administration ..............................................................................................49
  3.3 Engineering firm ...........................................................................................51
  3.4 Warehouse ....................................................................................................51
3.5 Roof department
3.5.1 Sawing step
3.5.2 Preparing step
3.5.3 Pre-montage step
3.5.4 Roof material step
3.5.5 Packing step of roof department

3.6 Window department
3.6.1 Sawing step
3.6.2 Subdivision for fixed windows and doors
3.6.3 Subdivision for sliding windows and doors
3.6.4 Packing step of window department

3.7 Accessories department

3.8 Transport

Chapter 4 Waste Reduction Problem

4.1 The Cutting Stock Problem: an introduction
4.2 Review of the literature
4.3 Classification of Cutting and Packing Problems
4.4 Approach of the waste reduction problem

4.5 The Cutting Stock Problem
4.5.1 An example
4.5.2 Mathematical model
4.5.3 Delayed column generation
4.5.4 Procedure

4.6 Improvement of column generation algorithm
4.6.1 The Cutting Stock Problem with low demands
4.6.2 Low Demand Column Generation Algorithm

4.7 AMPL program
4.7.1 csp.run file
4.7.2 csp.mod file
4.7.3 csp.dat file
4.7.4 Result

4.8 Discussion of specific CSP for the veranda industry

4.9 Exact Algorithm

Chapter 5 Company Software

5.1 Introduction to lp_solve
5.2 VB-code for low demand column generation algorithm
5.2.1 Adaptation of the algorithm
5.2.2 Library call and declaration
5.2.3 Reading input data from Excel
5.2.4 Defining master problem
5.2.5 Defining knapsack problem
5.2.6 The Cutting Stock Algorithm
5.2.7 Writing the solution to Excel
5.2.8 Eliminating excess of cut pieces
5.2.9 Illustration of Visual Basic program

5.3 VB-code for exact algorithm
5.3.1 Generating all ‘full’ and ‘required’ patterns
List of Figures

Figure 1-1  Veranco, the company ____________________________________________________________ 4
Figure 1-2  An example of the construction the company is making __________________________________________ 6
Figure 1-3  Another view of the construction the company is making ___________________________________________ 8
Figure 1-4  Veranda drawing with illustration of technical jargon __________________________________________ 11
Figure 1-5  Top and side view of the Agate veranda ____________________________________________________ 16
Figure 1-6  Top and side view of the Topaze veranda ____________________________________________________ 17
Figure 1-7  Top and side view of the Grenat veranda ____________________________________________________ 17
Figure 1-8  Top and side view of the Emeraude veranda __________________________________________________ 18
Figure 1-9  Top and side view of the Diamant veranda ___________________________________________________ 19
Figure 1-10  Top and side view of the Victorienne veranda _______________________________________________ 20
Figure 1-11  Top and side view of the Rubis veranda _____________________________________________________ 21
Figure 1-12  Polycarbonate ____________________________________________________________________________ 22
Figure 1-13  Sandwich panel ___________________________________________________________________________ 22
Figure 1-14  Double glazing ____________________________________________________________________________ 22

Figure 2-1  P-Q graph of the Roof Profile Types _______________________________________________________ 31
Figure 2-2  P-Q graph of the Thermal Break _____________________________________________________________ 32
Figure 2-3  P-Q graph of the Standard Shapes ___________________________________________________________ 33
Figure 2-4  P-Q graph of the Roofing Materials ___________________________________________________________ 34
Figure 2-5  P-Q graph of the Polycarbonate 25mm ________________________________________________________ 35
Figure 2-6  P-Q graph of the Sandwich panel _____________________________________________________________ 36
Figure 2-7  P-Q graph of the Colours ______________________________________________________________________ 37

Figure 3-1  Engineering Firm ____________________________________________________________ 51
Figure 3-2  Engineering Firm close-up _______________________________________________________________ 51
Figure 3-3  Stock profiles arriving ________________________________________________________________ 52
Figure 3-4  Fork-lift places profiles in warehouse _______________________________________________________ 52
Figure 3-5  Control of lacquer __________________________________________________________________________ 52
Figure 3-6  Stock profiles in warehouse _______________________________________________________________ 52
Figure 3-7  Overview of Roof Department 1 ____________________________________________________________ 53
Figure 3-8  Overview of Roof Department 2 ____________________________________________________________ 53
Figure 3-9  Roof profile on sawing machine ____________________________________________________________ 53
Figure 3-10  Sawing a roof profile ________________________________________________________________ 53
Figure 3-11  Cart with roof profiles ________________________________________________________________ 54
Figure 3-12  Placing rubber ____________________________________________________________________________ 54
Figure 3-13  Making a spotlight hole _________________________________________________________________ 54
Figure 3-14  Putting reinforcements _________________________________________________________________ 54
Figure 3-15  Pre-assembling the roof _________________________________________________________________ 55
Figure 3-16  Pre-montage step 1 _________________________________________________________________ 55
Figure 3-17  Pre-montage step 2 _________________________________________________________________ 55
Figure 3-18  Pre-montage step 3 _________________________________________________________________ 55
Figure 3-19  Stock of polycarbonate _________________________________________________________________ 56
Figure 3-20  Polycarbonate placed on a roof ____________________________________________________________ 56
Figure 3-21  Cutting a polycarbonate plate ______________________________________________________________ 56
Figure 3-22  Covering the edges with tape ______________________________________________________________ 56
Figure 3-23  Packing profiles 1 _________________________________________________________________ 57
Figure 3-24  Packing profiles 2 _________________________________________________________________ 57
Figure 3-25  Placing profiles in a carton _________________________________________________________________ 58
Figure 3-26  Sticking a sticker on the carton ____________________________________________________________ 58
Figure 3-27  Overview of window department 1 _________________________________________________________ 57
Figure 3-28  Overview of window department 2 _________________________________________________________ 57
Figure 3-29  Inputting the dimensions ________________________________________________________________ 58
Figure 3-30  Sawing window profiles _________________________________________________________________ 58
Figure 3-31  Placing the veranda lock ________________________________________________________________ 59
Figure 3-32  Boring the lock hole _________________________________________________________________ 59
List of Tables

Table 1-1 Technical Jargon ________________________________________________________________ 10
Table 2-1 Sales data 2005, Veranda Type _____________________________________________________ 24
Table 2-2 Sales data 2005, Roofing material ________________________________________________ 28
Table 2-3 Sales data 2005, Colour ________________________________________________________ 29
Table 2-4 P-Q analysis of the Roof Profile Types ____________________________________________ 31
Table 2-5 P-Q analysis of the Thermal Break _______________________________________________ 32
Table 2-6 P-Q analysis of the Standard Shapes ______________________________________________ 33
Table 2-7 P-Q analysis of the Roofing Materials ____________________________________________ 34
Table 2-8 P-Q analysis of the Polycarbonate 25mm ___________________________________________ 35
Table 2-9 P-Q analysis of the Sandwich Panel ______________________________________________ 36
Table 2-10 P-Q analysis of the Colours ____________________________________________________ 37
Table 4-1 Roof profile list ready to be optimized ____________________________________________ 71
Table 4-2 Example 1 CSP ________________________________________________________________ 71
Table 4-3 Example 2 CSP ________________________________________________________________ 72
Table 4-4 Repetition of example 1 CSP ____________________________________________________ 78
Table 4-5 Example 3 CSP ________________________________________________________________ 84
Table 5-1 Repetition of example 1 CSP ____________________________________________________ 108
Table 5-2 Output before a click on the solve button __________________________________________ 110
Table 5-3 Output after a click on the solve button ___________________________________________ 110
Table 5-4 Example 3 CSP ________________________________________________________________ 111
Table 5-5 Example 4 CSP ________________________________________________________________ 111
Table 5-6 Output of example 3 without removing the excessive pieces __________________________ 112
Table 5-7 Output of example 3 with removing the excessive pieces _____________________________ 112
Table 5-8 Output of example 4 without removing the excessive pieces __________________________ 113
Table 5-9 Output of example 4 with removing the excessive pieces _____________________________ 113
Table 5-10 Illustrative example of exact algorithm __________________________________________ 120
Table 5-11 Values of all counters and arrays displayed step-by-step ____________________________ 122
Table 5-12 Comparison results with optimizing program and without __________________________ 124
Veranco is a company producing customized verandas, ‘prêt-àposer’, in aluminium or wood. Verandas are also known as conservatories in British English or sunrooms in American English. Two problems, which the company is facing nowadays, will be discussed. The first problem is a cutting stock problem. When sawing the seven-meter aluminium stock profiles to the desired length, the company obtains a huge amount of waste material. The company estimates the waste at 25%. We will develop a program that can directly be implemented and run before starting a sawing batch, so that the waste is minimized. The second problem we will deal with is a design problem faced by the company, while cutting the angles of rising gutters.

In Chapter 1, the company, its history, its organization, and its product are presented. An explanation of the typical veranda jargon is given, which is clarified with a general veranda drawing. Next, the product itself is studied and subdivided in different product families. In the last part of this chapter, all subdivisions of the aluminium verandas are talked through.

Chapter 2 deals with the Product - Quantity analysis. First, we explain how the required data for the P-Q analysis were extracted from the company sales data. Subsequently, the P-Q tables and graphs are interpreted. Finally, an elaborate example of a veranda project together with the associated Bill Of Material (BOM) is given.

Chapter 3 is devoted to the veranda production process, from the moment the order is received towards the transportation to the client. The process consists of the following global steps: an administration step, an engineering step, a manufacturing step and the transportation to the client. The fabrication is done in three departments: the roof department, the window department and the accessories department.
Introduction

In Chapter 4, the Cutting Stock Problem (CSP) is discussed. First, an introduction about the CSP and a review of the literature are given. Next, a classification of cutting and packing problems is made. Thereupon, Gilmore and Gomory’s algorithm for the CSP is expounded. To minimize the waste material, the veranda orders are first analysed and regrouped into packages of the same profile and colour. Subsequently, an algorithm is run for each package. Two algorithms are developed. The first algorithm is an adaptation of Gilmore and Gomory’s algorithm for low demands, i.e. a low demand column generation algorithm. The second algorithm is an exact algorithm, where the patterns are generated on a smart way.

In Chapter 5, we present two easy-to-use programs to reduce the waste material of the company. The programs are written in Visual Basic. The CSP is solved utilizing lp_solve, an open-source Mixed Integer Linear Programming (MILP) solver based on the revised simplex method and the Branch-and-Bound method for integer values. The first program is based on the low demand column generation algorithm and the second program is based on the exact algorithm. Finally, the results are compared with the situation without the optimizing program using real data of the company.

Chapter 6 deals with the design problem. In the case of a rising gutter, a combination of a vertical and horizontal cutting angle has to be sawed. These angles are not just the half of the vertical and horizontal angles of the rising gutter. A model with 100 x 100 beams is constructed to clarify the 3D visualisation. Subsequently, the vertical cutting angle, the torsion angle and the horizontal cutting angle are calculated in function of the given vertical and horizontal angle of the rising gutter.
Chapter 1

Product Overview

In this chapter first, a little word will be said about the company, its history, its organization. Thereupon the word ‘veranda’ will be elucidated. We will research the application of the word ‘veranda’ in four different languages: English, Dutch, French and Spanish. Next, a veranda vocabulary will be built up in the same four languages. A general veranda drawing clarifies the technical jargon. Finally, the product itself is studied and subdivided in different product families. All subdivision of the aluminium verandas are talked through.

1.1 The company

Veranco is a company producing customized verandas, in aluminium or wood, ready to erect, or ‘prêt-à-poser’ like they call it. The company is located in Sint-Niklaas (Belgium), between Ghent and Antwerp (Figure 1.1). Veranco is the first veranda network in Europe, present in France, Switzerland, Netherlands, Belgium and Luxemburg.

All the veranda projects are completely manufactured in the factory, pre-assembled to the millimetre and delivered to the client accompanied with all necessary supplies: inox fastenings, neoprene joints, glue … All the non-rectangular and non-flat roofed verandas are pre-mounted in the factory to ensure an easy installation.

Today the veranda network counts more than 60 centres in France, Switzerland and the Benelux. Every Veranco centre is managed by an independent partner, responsible for the sales, promotion and erection, i.e. build up of the verandas in his own sector.
Veranco was created on March, 1st 1983 with 3 factory workers, two employees and one responsible. For many years the turnover or amount of total sales increased 15 to 20 % yearly. Nowadays every 26 minutes 1 veranda is produced.

Today the company has a strong brand name and has found his cruising speed with a turnover of 13,5 millions €. This turnover is realised with 115 people, all active in Sint-Niklaas. The fabrication unit consists of two overseers, 11 line chiefs and 75 specialised workers. In the offices 19 employees are busy running the enterprise:

- 1 responsible for the engineering firm in which 12 engineers are preparing the construction of the verandas
- 1 in charge of the logistics organises the transports realised by international transporters
- 1 responsible for the quality is busy with the fabrication improvement and is in charge of all the complaints
- 1 IT specialist, in charge of the computer network and all data communication
Chapter 1: Product Overview

- 1 responsible for the R&D develops the new products while listening to the clients desires
- 1 HR manager, in charge of recruiting people
- 1 accountant

The board of directors, consisting of one chief executive officer (CEO), one commercial director and one technical director assisted by their secretaries, manage the whole team. The veranda distribution is realised through a distribution network. The distributors (partners) are independent enterprises fixed on the commercial aspect. They manage a sales team, do commercial actions like fairs and all have a nice exhibition. They are also responsible for the erection, i.e. build up, of the verandas in their own sector and the after sales service.

Veranco also trains the distributors. Trainings are organised every month to train the chiefs, the salespersons, the people who take the dimensions and the ones who build up the verandas.

### 1.2 What’s a veranda?

Although we find in the dictionary as the translation of the Dutch word ‘veranda’, veranda or verandah in English and porch in American English, the word veranda in English is not as common as it is in Dutch and French (véranda). In Spanish we find the same difficulty, veranda can be found in the Diccionario de la lengua española (Real Academia Española) but is not common use. More common are galería or porche, or eventually galería tipo veraneo, construcción en vidrio or construcción veranea.

The best way to describe the construction we want to talk about is simply a picture … (Figure 1.2).
A **verandah** is a large porch or balcony, usually roofed and often partly enclosed, extending along the outside of a building, and also called regionally gallery. A verandah may encompass the entire façade as well as the sides of a structure.

Its origin can be traced to a Hindi and Bengali word ‘barandah’, which refers to the open area around one's house. A verandah can either be full width of the frontage, or extend around the sides and sometimes rear of the building. The word ‘verandah’ is also a common word in Malayalam (Spoken in Kerala, India). A verandah is a part of traditional architecture of Kerala. It is an open balcony in the front side or around the main structure, supported by pillars.

The name ‘Veranda’ is also a brand of composite decking and railing which is a wood-alternative offering the look of wood without the drawbacks of corroding, splintering, termite damage, or decay. It is a composite material composed of recycled wood and plastic.
A **porch** is an architectural feature relating to a floor-like platform structure attached to the front or back entrance of a residence. It is external to the walls of the main building proper, but it may be enclosed by screen, latticework, broad windows, or other light frame walls extending from the main structure. The porch serves as a place to pause comfortably before entering or exiting.

The porch, especially in the southern United States, is often as broad as it is deep, and may provide enough space for residents to entertain guests or gather on special occasions. However, many American homes built since the 1940s with a porch only have a token one, i.e. a symbolic one, too small for comfortable social use and adding only to the visual impression of the building.

When it is covered, not only it provides protection from sun or rain, but it may also form, in effect, an extra exterior room that may accommodate chairs, tables and other furniture, to be used as a living space.

Notwithstanding the fact that we find the words veranda(h) and porch in every Dutch ↔ English or French ↔ English dictionary, they are not used a lot for buildings as you can see in the pictures (Figures 1.2 and 1.3). Words like conservatory and sunroom are more common. But looking in the dictionary, we see that these words are in the first place used for large greenhouses.

A conservatory is a large greenhouse, especially one in which plants are arranged aesthetically for display, as at a botanical garden. It is a glass and metal structure traditionally found in the gardens of large houses. Modern versions are smaller, can be made of PVC and are often added to houses for home improvement purposes. Conservatories can be both greenhouses and recreational spaces. The modern domestic conservatory is used as an extra room rather than for horticulture.

A solarium is a room largely built of glass to afford exposure to the sun. Sometimes known as a porch enclosure, it has window walls and a roof of either solid foam-insulated panels or glass.
Although both words are often used to describe the same structure, the term ‘conservatory’ usually refers to vinyl-framed structures with pitched roofs. ‘Sunroom’ usually refers to solid-wood-framed structures in studio or shed-roof styles, finally ‘conservatory’ tends to be very much a British term, while ‘sunroom’ is a term more frequently used in the USA and Canada.

Besides conservatory and sunroom, other words like ‘patio enclosure’, ‘patio room’, ‘sun porch’, ‘glassed-in porch’ and ‘garden room’ are also used to depict the same construction.

In this work the word ‘veranda’ will be used as it is close to the English word ‘veranda(h)’, the French word ‘véranda’ and similar to the Dutch and Spanish word ‘veranda’. ‘Veranda’ has not completely the same meaning in every language and is not used a lot in every language, but it is a good global approximation.
1.3 Veranda vocabulary

This section gives the translation of words that will be used in the course of this work. Let us say some typical veranda jargon. The translation was made in four languages as the company’s first language is French and this work was made partly in Belgium and partly in Spain. After the vocabulary table, some words are explained by a picture for a better understanding.
<table>
<thead>
<tr>
<th><strong>FRENCH</strong></th>
<th><strong>DUTCH</strong></th>
<th><strong>ENGLISH</strong></th>
<th><strong>SPANISH</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>chevron intermediaire</td>
<td>midden dakligger</td>
<td>intermediate rafter</td>
<td>limatesa</td>
</tr>
<tr>
<td>faîtière</td>
<td>muurprofiel</td>
<td>ridge-tile</td>
<td>cumbrera</td>
</tr>
<tr>
<td>chêneau</td>
<td>dakgoot</td>
<td>gutter</td>
<td>cañería</td>
</tr>
<tr>
<td>triangle de rive</td>
<td>zijkant driehoek</td>
<td>side triangle</td>
<td>triángulo lateral</td>
</tr>
<tr>
<td>chevron lateral</td>
<td>zijdkligger</td>
<td>lateral rafter</td>
<td>viga lateral</td>
</tr>
<tr>
<td>menuiseries</td>
<td>ramen en deuren</td>
<td>windows and doors</td>
<td>ventanas y puertas</td>
</tr>
<tr>
<td>descente d'eau pluviale</td>
<td>regenpijp</td>
<td>rainwater pipe</td>
<td>desagüe</td>
</tr>
<tr>
<td>traverse</td>
<td>dwarsbalk</td>
<td>transom (crossbeam)</td>
<td>travesaño</td>
</tr>
<tr>
<td>allège</td>
<td>steunmuur (onder venster)</td>
<td>retaining wall</td>
<td>antepecho</td>
</tr>
<tr>
<td>serrure à cylindre</td>
<td>cilinderslot</td>
<td>cylinder lock</td>
<td>cerradura cilíndrica</td>
</tr>
<tr>
<td>ouvrant à la francaise</td>
<td>opendraaiend raam</td>
<td>sliding window</td>
<td>ventana corredera</td>
</tr>
<tr>
<td>rupture de pont thermique</td>
<td>thermische onderbreking</td>
<td>thermal break</td>
<td>aislamiento térmico</td>
</tr>
<tr>
<td>tabatière</td>
<td>dakvenster</td>
<td>luthern</td>
<td>buhardilla</td>
</tr>
<tr>
<td>chassis coulissant</td>
<td>schuifdeur (raam)</td>
<td>sliding door (window)</td>
<td>puerta corredera</td>
</tr>
<tr>
<td>facade</td>
<td>facade, voorkant</td>
<td>facade, front</td>
<td>fachada</td>
</tr>
<tr>
<td>crêpine</td>
<td>zuigkorf, filter</td>
<td>inlet filter</td>
<td>filtro</td>
</tr>
</tbody>
</table>

*Table 1-1 Technical Jargon*
Figure 1-4 Veranda drawing with illustration of technical jargon
1.4 Product families

There are two big classes of products: *grilles optimisées* (standard verandas) and *création sur mesure* (custom-built verandas). As the second group is composed of special verandas, very personally, depending on the client, we will only deal with the first group.

- The first subdivision should be between aluminium and wooden verandas. There are two types of wooden verandas: Pergolas and Perlandas. Pergolas are completely wooden verandas made of ‘Bankirai’, sort of wood from Borneo with a very high density, more than 1000 kg/m³. Perlandas are a mix of wooden and aluminium structures. These wooden verandas are strongly different from the aluminium ones, because they are much less important (see the P-Q Analysis in Chapter 2) we will not speak more about this type of verandas in this work.

- The second and most important subdivision are the roof profile types. In the company they are divided in three kinds: Ariane, Confort and Prestige+. There are two types of window and door frames: Confort and Prestige+. The Ariane roof profiles goes together with Confort window frames. The Confort and Prestige+ roof profiles obviously go together with the respectively the Confort and Prestige+ window frames (see Section 1.4.1).

- In the *grilles optimisées* (standard shapes) there are seven facades or standard shapes: agate, topaze, grenat, emeraude, diamant, victorienne and rubis (see Section 1.4.2).

- A following subdivision can be made for the three roofing materials: polycarbonate, sandwich panel and double glazing (see Section 1.4.3).

- The last subdivision is for the colour. There are three basic colours which are all in stock: white ‘blanc neige’ (RAL9010), chestnut ‘marron normand’ (RAL 8019) and beige ‘beige ton pierre’ (RAL1015). The numbers between brackets refer to the RAL colour space system. Besides this, there are six frequently used lacquered colours ‘LC 6’, nine lacquered matt colours ‘LMAT9’ and a lot of less used colours. It is also possible to combine two colours in a veranda: ‘bicolor’ if profiles with two different colours are used and ‘duocolor’ if a mix of profiles of two different colours are used.

The above subdivisions are graphically displayed in the following diagrams. The RAL colours were converted to RGB colours to show the exact colour on paper, using following site: [http://formulaire.pats.ch/~form/fr/unites/unites11.asp](http://formulaire.pats.ch/~form/fr/unites/unites11.asp).
1.4.1 Roof profile types

1.4.1.1 Ariane

This is the basic roof type for smaller budgets or limited depth. Some dimensional characteristics are:

- Maximum depth is 2’90m much less than 4’00m for the Confort and Prestige+.
- Maximum height of the windows is 2’00m instead of 2’10m.

A detailed plan of the used profiles can be consulted in Appendix B.

1.4.1.2 Confort

This is the middle class, with a thermal interruption bridge for following profiles: the ridge-tile, the gutter and the intermediate rafter. So there is no thermal interruption bridge for the lateral rafter. The thermal break provides a layer of insulation that will resist heat flow through the thermal bridge, and has therefore a positive influence on the condensation. A detailed plan of the used profiles can be consulted in Appendix B.

1.4.1.3 Prestige+

The Prestige+ veranda is the high-end product line, with a high quality and aestheticism. All the profiles have a thermal interruption bridge. These thermal interruption bridges are realised with rods reinforced with glass. The isolation quality is close to this of double glass. Condensation risks on the gutter, the rafters, the ridge-tile and the side triangles are considerably diminished. There are to types of Prestige+, the Prestige+ M and the Prestige+ XL, which is bigger and stronger, designed for roofs which have to resist strong wind or big mass of snow. A detailed plan of the used profiles can be consulted in Appendix B.
1.4.2 Standard shapes

This section gives an overview of all standard shapes. For each shape, a little description is given accompanied with a top and a side view of the veranda.

1.4.2.1 Agate

The Agate veranda has a rectangular roof with gutter only in the façade. Ventilation is provided high in the front windows. Front windows/doors are always sliding ones with minimum 1 fixed one. The two corner posts (90°) are foreseen of a raining pipe (Ø 80 mm) with inlet filter, integrated in the post structure. The side triangle is made of double glass.

![Figure 1-5 Top and side view of the Agate veranda](image)

1.4.2.2 Topaze

The Topaze veranda has a rectangular roof with different facets, i.e. planes. The gutter is all along the veranda. Front windows/doors are always sliding ones with minimum 1 fixed one. Ventilation is provided high in the front windows. The two corner posts (90°) are foreseen of a raining pipe (Ø 80 mm) with inlet filter, integrated in the post structure. There are no side triangles, as the upper side frames are not parallel with the lower side frames, but have an inclination towards the middle of the veranda.
1.4.2.3 Grenat

The Grenat veranda has a rectangular roof with different facets and a double slope. The gutter is all along the veranda. Front windows/doors are always sliding ones with minimum 1 fixed one. Ventilation is provided high in the front windows. The two corner posts (90°) are foreseen of a raining pipe (⌀ 80 mm) with inlet filter, integrated in the post structure. There are no side triangles, as the upper side frames are not parallel with the lower side frames, but have an inclination towards the middle of the veranda.
1.4.2.4 Emeraude

The Emeraude veranda has a roof shape as in picture …. The roof has different facets, as can be seen in the side-view of the veranda. The gutter is all along the veranda. Front windows/doors are always sliding ones with minimum 1 fixed one. On each side a single cut made by a fixed window of 1134 mm length that makes a 45° angle. Ventilation is also provided in these fixed windows. The four corner posts (135°) are foreseen of a raining pipe (Ø 80 mm) with inlet filter, integrated in the post structure. There are no side triangles, as the upper side frames are not parallel with the lower side frames, but have an inclination towards the middle of the veranda. An outside slope of more than 10° is advised for a better aesthetic.

Figure 1-8  Top and side view of the Emeraude veranda
1.4.2.5 Diamant

The Diamant veranda has a roof shape as in picture .... The roof has different facets, as can be seen in the side-view of the veranda. The gutter is all along the veranda. Front windows/doors are fixed ones for small dimensions and sliding for big dimensions. On each side a double cut made by fixed windows of 1101 mm and 1107 mm length. Ventilation is also provided in these fixed windows. The six corner posts (135°) are foreseen of a raining pipe (⌀ 80 mm) with inlet filter, integrated in the post structure. There are no side triangles, as the upper side frames are not parallel with the lower side frames, but have an inclination towards the middle of the veranda. An outside slope of more than 10° is advised for a better aesthetic.

Figure 1-9 Top and side view of the Diamant veranda
1.4.2.6 Victorienne

The Victorienne veranda has the particularity of being composed of a radiant part on the front side and of a second part with a double slope at the back. For the radiant part there are four different modules each with five faces. All frames are fix ones. In two of them ventilation is provided. The second part is variable between 0 m and 4 m. The four corner posts (135°) are foreseen of a raining pipe (Ø 80 mm) with inlet filter, integrated in the post structure.

Figure 1-10 Top and side view of the Victorienne veranda
1.4.2.7 Rubis

The Rubis veranda has a roof with different facets and a double slope as can be seen in the side-view of the veranda. The gutter is all along the veranda. Front windows/doors are always sliding ones with minimum 1 fixed one. On each side a single cut made by a fixed window of 1134 mm length that makes a 45° angle. Ventilation is also provided in these fixed windows. There are no side triangles, as the upper side frames are not parallel with the lower side frames, but have an inclination towards the middle of the veranda. An outside slope of more than 10° is advised for a better aesthetic.

Figure 1-11  Top and side view of the Rubis veranda
Chapter 1  Product Overview

1.4.3 Roofing materials

1.4.3.1 Polycarbonate

Polycarbonate is used a lot as roofing material due to its strong mechanic resistance, light weight, thermal resistance and light transmission properties. This material has an excellent isolation coefficient due to the multiple partitions: 4, 5 or 6 depending on the thickness 16 mm, 25 mm or 35 mm, respectively.

Figure 1-12 Polycarbonate

1.4.3.2 Sandwich panel

This roofing material is so called because it is made of two aluminium sheets and between them polystyrene. The aluminium provides the necessary rigidity, and the polystyrene, the thermal isolation. There are three different thicknesses: 25, 42 and 92 mm.

Figure 1-13 Sandwich panel

1.4.3.3 Double glazing

Double glazing gives maximal profit of the nature environment. It consists of an outside glass with or without isolation layer and an inside laminated glass to keep glass pieces on their place in case of break.

Figure 1-14 Double glazing
Chapter 2

Data Extraction and P-Q Analysis

The first part of this chapter explains how the required data for the Product – Quantity analysis (P-Q Analysis) were extracted from the company sales data. The raw data were studied and regrouped according to the required P-Q analysis. The sales of similar entities were added together in Excel sheets to make graphs of them. Subsequently, in the second part of this chapter the P-Q graphs, made over a period of five years are interpreted. The third part gives an elaborate example of a veranda project and the associated Bill Of Material (BOM), to get a good view of what they are doing in the engineering firm.

2.1 Data extraction

To make a P-Q analysis we first have to extract the required data from the raw data. The enterprise provided the sales data from the past five years, so they were available for analysing purpose. In this section, the sales data of 2005 will be used to show how the data extraction was made starting with the received raw data. This raw data consist of a couple of lists with abbreviations and numbers.

2.1.1 Roof profile type, thermal break and standard shape

To extract roof profile type, thermal break and standard shape, the following list is used:

<table>
<thead>
<tr>
<th>Type</th>
<th>Points</th>
<th>Number</th>
<th>Mean</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIV</td>
<td>731</td>
<td>42</td>
<td>17.39</td>
<td>0.38</td>
</tr>
<tr>
<td>MC</td>
<td>517</td>
<td>37</td>
<td>13.97</td>
<td>0.27</td>
</tr>
<tr>
<td>MC+P</td>
<td>1678</td>
<td>65</td>
<td>25.82</td>
<td>0.86</td>
</tr>
<tr>
<td>MP</td>
<td>693</td>
<td>29</td>
<td>23.9</td>
<td>0.36</td>
</tr>
</tbody>
</table>
Table 2-1  Sales data 2005, Veranda Type

<table>
<thead>
<tr>
<th>Veranda Type</th>
<th>Points</th>
<th>Produced Verandas</th>
<th>Price</th>
<th>Global Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP+P</td>
<td>2,682</td>
<td>61</td>
<td>43,96</td>
<td>1,38</td>
</tr>
<tr>
<td>PG</td>
<td>691</td>
<td>13</td>
<td>53,14</td>
<td>0,36</td>
</tr>
<tr>
<td>PGO</td>
<td>994</td>
<td>19</td>
<td>52,31</td>
<td>0,51</td>
</tr>
<tr>
<td>PLC</td>
<td>476</td>
<td>4</td>
<td>118,97</td>
<td>0,24</td>
</tr>
<tr>
<td>PLCA</td>
<td>1,093</td>
<td>11</td>
<td>99,39</td>
<td>0,56</td>
</tr>
<tr>
<td>PLCE</td>
<td>683</td>
<td>5</td>
<td>136,56</td>
<td>0,35</td>
</tr>
<tr>
<td>PLP</td>
<td>1,068</td>
<td>8</td>
<td>133,52</td>
<td>0,55</td>
</tr>
<tr>
<td>PLPE</td>
<td>611</td>
<td>3</td>
<td>203,64</td>
<td>0,31</td>
</tr>
<tr>
<td>PLPT</td>
<td>370</td>
<td>3</td>
<td>123,47</td>
<td>0,19</td>
</tr>
<tr>
<td>TA</td>
<td>3,474</td>
<td>98</td>
<td>35,45</td>
<td>1,79</td>
</tr>
<tr>
<td>TC</td>
<td>23,249</td>
<td>347</td>
<td>67</td>
<td>11,95</td>
</tr>
<tr>
<td>TP</td>
<td>40,541</td>
<td>322</td>
<td>125,9</td>
<td>20,84</td>
</tr>
<tr>
<td>TP+</td>
<td>14,362</td>
<td>106</td>
<td>135,49</td>
<td>7,38</td>
</tr>
<tr>
<td>VAE</td>
<td>4,569</td>
<td>105</td>
<td>43,51</td>
<td>2,35</td>
</tr>
<tr>
<td>VCA</td>
<td>11,675</td>
<td>164</td>
<td>71,19</td>
<td>6,00</td>
</tr>
<tr>
<td>VCD</td>
<td>2,041</td>
<td>18</td>
<td>113,41</td>
<td>1,05</td>
</tr>
<tr>
<td>VCE</td>
<td>15,408</td>
<td>166</td>
<td>92,82</td>
<td>7,92</td>
</tr>
<tr>
<td>VCG</td>
<td>430</td>
<td>5</td>
<td>85,98</td>
<td>0,22</td>
</tr>
<tr>
<td>VCR</td>
<td>2,286</td>
<td>22</td>
<td>103,92</td>
<td>1,18</td>
</tr>
<tr>
<td>VCT</td>
<td>5,456</td>
<td>63</td>
<td>86,6</td>
<td>2,80</td>
</tr>
<tr>
<td>VCV</td>
<td>1,307</td>
<td>11</td>
<td>118,78</td>
<td>0,67</td>
</tr>
<tr>
<td>VPA</td>
<td>6,940</td>
<td>74</td>
<td>93,78</td>
<td>3,57</td>
</tr>
<tr>
<td>VPA+</td>
<td>2,357</td>
<td>25</td>
<td>94,26</td>
<td>1,21</td>
</tr>
<tr>
<td>VPAS</td>
<td>678</td>
<td>8</td>
<td>84,74</td>
<td>0,35</td>
</tr>
<tr>
<td>VPD</td>
<td>4,427</td>
<td>29</td>
<td>152,64</td>
<td>2,28</td>
</tr>
<tr>
<td>VPD+</td>
<td>1,858</td>
<td>12</td>
<td>154,8</td>
<td>0,95</td>
</tr>
<tr>
<td>VPDS</td>
<td>317</td>
<td>2</td>
<td>158,3</td>
<td>0,16</td>
</tr>
<tr>
<td>VPE</td>
<td>17,139</td>
<td>140</td>
<td>122,42</td>
<td>8,81</td>
</tr>
<tr>
<td>VPE+</td>
<td>6,042</td>
<td>49</td>
<td>123,31</td>
<td>3,11</td>
</tr>
<tr>
<td>VPES</td>
<td>1,267</td>
<td>8</td>
<td>158,31</td>
<td>0,65</td>
</tr>
<tr>
<td>VPG</td>
<td>1,232</td>
<td>10</td>
<td>123,24</td>
<td>0,63</td>
</tr>
<tr>
<td>VPG+</td>
<td>324</td>
<td>3</td>
<td>108,05</td>
<td>0,17</td>
</tr>
<tr>
<td>VPR</td>
<td>3,384</td>
<td>25</td>
<td>135,36</td>
<td>1,74</td>
</tr>
<tr>
<td>VPR+</td>
<td>1,197</td>
<td>9</td>
<td>133,01</td>
<td>0,62</td>
</tr>
<tr>
<td>VPRS</td>
<td>377</td>
<td>2</td>
<td>188,67</td>
<td>0,19</td>
</tr>
<tr>
<td>VPT</td>
<td>5,217</td>
<td>47</td>
<td>111,01</td>
<td>2,68</td>
</tr>
<tr>
<td>VPT+</td>
<td>1,787</td>
<td>16</td>
<td>111,67</td>
<td>0,92</td>
</tr>
<tr>
<td>VPTS</td>
<td>378</td>
<td>3</td>
<td>126,09</td>
<td>0,19</td>
</tr>
<tr>
<td>VPV</td>
<td>1,321</td>
<td>9</td>
<td>146,76</td>
<td>0,68</td>
</tr>
<tr>
<td>VPV+</td>
<td>1,239</td>
<td>10</td>
<td>123,91</td>
<td>0,64</td>
</tr>
</tbody>
</table>

This list gives information about the veranda types (basic material, roof profile type, frame type and shape). In the first column the name of the veranda type is given. The company works with points instead of a currency unit, which can be seen in the second column. The third column gives the number of produced verandas of each type in 2005. The fourth column is the second one divided by the third one. The last column gives the global percentage. For the analysis we will work with points instead of produced verandas as bigger verandas, which will count for more points, are more important than smaller verandas.
The first letter of the veranda type gives basic information. The second letter gives information about the profile type and depends on the first letter.

The letters stand for:

**DIV** = Divers (various)

**M** = Menuiserie (windows and doors)

- **C** = Confort
- **P** = Prestige

**T** = Toiture (roof)

- **A** = Ariane
- **C** = Confort
- **P** = Prestige

**P** = Wooden verandas

- **G** = Pergola
- **L** = Perlanda

  - **C** = Confort
  - **P** = Prestige

**V** = Veranda

- **A** = Ariane
- **C** = Confort
- **P** = Prestige

With these two letters the analysis ‘roof profile type’ and ‘thermal break’ can be made. For example, for the Confort Type we add up all the points from any reference where the second letter is ‘C’. This gives: \(23249 + 11675 + 2041 + 15408 + 430 + 2286 + 5456 + 1307 = 61852\). The same thing can be done for the Prestige+ (P) and Ariane (A). All names starting with an ‘M’ are not complete verandas, but only doors and windows (MS), and showed as MS in the ‘roof profile’ graph to make contrast with the complete verandas (VA, VC, VP). In the ‘thermal break’ analysis MC and MP are separated, as there is no thermal break in the first one and there is in the second one. \((MS = MC + MP)\) The same thing can be said for the wooden verandas: PLP is with thermal break; PLC and PG without.
The third letter gives information about the veranda shape and will be used to make the ‘standard shape’ analysis. This analysis was only done for aluminium verandas. The third letter is always the first letter of one of the different shapes: agate (A), topaze (T), grenat (G), emeraude (E), diamante (D), victorienne (V) and rubis (R). For example, to be aware of the importance of the Agate shape we add up all verandas with an Agate shape: $11675 + 6940 + 2357 + 678 = 21649$.

### 2.1.2 Roof material

To make the ‘roof material’ analysis, the following raw data was used:

<table>
<thead>
<tr>
<th>Roof Material (main, secondary)</th>
<th>Points</th>
<th>Number</th>
<th>Mean</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>P16C</td>
<td>957</td>
<td>33</td>
<td>29.01</td>
<td>0.71</td>
</tr>
<tr>
<td>P16C,P16O</td>
<td>38</td>
<td>1</td>
<td>37.97</td>
<td>0.03</td>
</tr>
<tr>
<td>P16C,PS25BAA</td>
<td>39</td>
<td>1</td>
<td>39.29</td>
<td>0.03</td>
</tr>
<tr>
<td>P16O</td>
<td>2027</td>
<td>60</td>
<td>33.78</td>
<td>1.49</td>
</tr>
<tr>
<td>P16O,P25O</td>
<td>136</td>
<td>2</td>
<td>67.91</td>
<td>0.10</td>
</tr>
<tr>
<td>P25AT</td>
<td>32675</td>
<td>455</td>
<td>71.81</td>
<td>24.08</td>
</tr>
<tr>
<td>P25AT,P25B</td>
<td>121</td>
<td>1</td>
<td>121.42</td>
<td>0.09</td>
</tr>
<tr>
<td>P25AT,P25C</td>
<td>403</td>
<td>3</td>
<td>134.2</td>
<td>0.30</td>
</tr>
<tr>
<td>P25AT,PS25BA</td>
<td>632</td>
<td>7</td>
<td>90.35</td>
<td>0.47</td>
</tr>
<tr>
<td>P25AT,PS25BB</td>
<td>1319</td>
<td>13</td>
<td>101.47</td>
<td>0.97</td>
</tr>
<tr>
<td>P25AT,PS25BT</td>
<td>138</td>
<td>1</td>
<td>137.99</td>
<td>0.10</td>
</tr>
<tr>
<td>P25AT,SKY</td>
<td>269</td>
<td>2</td>
<td>134.66</td>
<td>0.20</td>
</tr>
<tr>
<td>P25B</td>
<td>6331</td>
<td>78</td>
<td>81.17</td>
<td>4.67</td>
</tr>
<tr>
<td>P25B,P25AT</td>
<td>63</td>
<td>1</td>
<td>63.49</td>
<td>0.05</td>
</tr>
<tr>
<td>P25B,P25C</td>
<td>71</td>
<td>1</td>
<td>71.14</td>
<td>0.05</td>
</tr>
<tr>
<td>P25B,P25O</td>
<td>90</td>
<td>1</td>
<td>90.03</td>
<td>0.07</td>
</tr>
<tr>
<td>P25B,PS25BB</td>
<td>199</td>
<td>2</td>
<td>99.56</td>
<td>0.15</td>
</tr>
<tr>
<td>P25B,STDV</td>
<td>100</td>
<td>1</td>
<td>99.73</td>
<td>0.07</td>
</tr>
<tr>
<td>P25C</td>
<td>4807</td>
<td>75</td>
<td>64.09</td>
<td>3.54</td>
</tr>
<tr>
<td>P25C,P25B</td>
<td>205</td>
<td>2</td>
<td>102.26</td>
<td>0.15</td>
</tr>
<tr>
<td>P25C,P25O</td>
<td>77</td>
<td>1</td>
<td>77.48</td>
<td>0.06</td>
</tr>
<tr>
<td>P25C,PS25BB</td>
<td>311</td>
<td>4</td>
<td>77.76</td>
<td>0.23</td>
</tr>
<tr>
<td>P25HS</td>
<td>21</td>
<td>1</td>
<td>20.97</td>
<td>0.02</td>
</tr>
<tr>
<td>P25O</td>
<td>5511</td>
<td>91</td>
<td>60.56</td>
<td>4.06</td>
</tr>
<tr>
<td>P25O,P25B,P2</td>
<td>21</td>
<td>1</td>
<td>21.1</td>
<td>0.02</td>
</tr>
<tr>
<td>P25O,P25C</td>
<td>213</td>
<td>2</td>
<td>106.39</td>
<td>0.16</td>
</tr>
<tr>
<td>P25O,PS25BB</td>
<td>282</td>
<td>3</td>
<td>93.91</td>
<td>0.21</td>
</tr>
<tr>
<td>P25O,PS25BBA</td>
<td>101</td>
<td>1</td>
<td>100.54</td>
<td>0.07</td>
</tr>
<tr>
<td>P25O,PS25BTA</td>
<td>39</td>
<td>1</td>
<td>38.92</td>
<td>0.03</td>
</tr>
<tr>
<td>P35RE</td>
<td>3084</td>
<td>32</td>
<td>96.38</td>
<td>2.27</td>
</tr>
<tr>
<td>P35RE,P25O</td>
<td>99</td>
<td>1</td>
<td>99.33</td>
<td>0.07</td>
</tr>
<tr>
<td>P35RE,PS25BB</td>
<td>80</td>
<td>1</td>
<td>79.93</td>
<td>0.06</td>
</tr>
<tr>
<td>PS25BAAC</td>
<td>9500</td>
<td>82</td>
<td>115.86</td>
<td>7.00</td>
</tr>
<tr>
<td>PS25BAAC,?</td>
<td>185</td>
<td>1</td>
<td>185.05</td>
<td>0.14</td>
</tr>
<tr>
<td>PS25BAAC,P25</td>
<td>308</td>
<td>4</td>
<td>76.88</td>
<td>0.23</td>
</tr>
<tr>
<td>PS25BAAC,PS4</td>
<td>297</td>
<td>1</td>
<td>297.19</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>--------</td>
<td>---</td>
<td>-----</td>
<td>--</td>
</tr>
<tr>
<td>PS25BAAC,SKY</td>
<td>4425</td>
<td>30</td>
<td>147,49</td>
<td>3,26</td>
</tr>
<tr>
<td>PS25BAAC,STD</td>
<td>3704</td>
<td>28</td>
<td>132,29</td>
<td>2,73</td>
</tr>
<tr>
<td>PS25BAAC,ZDV</td>
<td>896</td>
<td>8</td>
<td>111,95</td>
<td>0,66</td>
</tr>
<tr>
<td>PS25BAAC,ZSV</td>
<td>193</td>
<td>1</td>
<td>193,28</td>
<td>0,14</td>
</tr>
<tr>
<td>PS25BB</td>
<td>8726</td>
<td>114</td>
<td>76,55</td>
<td>6,43</td>
</tr>
<tr>
<td>PS25BB,P25AT</td>
<td>1418</td>
<td>14</td>
<td>101,25</td>
<td>1,04</td>
</tr>
<tr>
<td>PS25BB,P25B</td>
<td>26</td>
<td>1</td>
<td>25,79</td>
<td>0,02</td>
</tr>
<tr>
<td>PS25BB,P25C</td>
<td>234</td>
<td>2</td>
<td>117,11</td>
<td>0,17</td>
</tr>
<tr>
<td>PS25BB,P25O</td>
<td>185</td>
<td>1</td>
<td>185,24</td>
<td>0,14</td>
</tr>
<tr>
<td>PS25BB,P35RE</td>
<td>71</td>
<td>1</td>
<td>70,75</td>
<td>0,05</td>
</tr>
<tr>
<td>PS25BB,PS25B</td>
<td>200</td>
<td>1</td>
<td>200,23</td>
<td>0,15</td>
</tr>
<tr>
<td>PS25BB,PS42B</td>
<td>211</td>
<td>1</td>
<td>211,48</td>
<td>0,16</td>
</tr>
<tr>
<td>PS25BB,SKY</td>
<td>1587</td>
<td>12</td>
<td>132,22</td>
<td>1,17</td>
</tr>
<tr>
<td>PS25BB,STDV</td>
<td>2245</td>
<td>21</td>
<td>106,88</td>
<td>1,65</td>
</tr>
<tr>
<td>PS25BB,ZDV</td>
<td>518</td>
<td>5</td>
<td>103,6</td>
<td>0,38</td>
</tr>
<tr>
<td>PS25BBAC</td>
<td>7125</td>
<td>64</td>
<td>111,32</td>
<td>5,25</td>
</tr>
<tr>
<td>PS25BBAC,P25</td>
<td>1011</td>
<td>8</td>
<td>126,43</td>
<td>0,75</td>
</tr>
<tr>
<td>PS25BBAC,P35</td>
<td>208</td>
<td>2</td>
<td>104</td>
<td>0,15</td>
</tr>
<tr>
<td>PS25BBAC,SKY</td>
<td>2037</td>
<td>18</td>
<td>113,16</td>
<td>1,50</td>
</tr>
<tr>
<td>PS25BBAC,STD</td>
<td>3710</td>
<td>28</td>
<td>132,5</td>
<td>2,73</td>
</tr>
<tr>
<td>PS25BBAC,ZDV</td>
<td>287</td>
<td>3</td>
<td>95,79</td>
<td>0,21</td>
</tr>
<tr>
<td>PS25BEAC</td>
<td>2755</td>
<td>28</td>
<td>98,39</td>
<td>2,03</td>
</tr>
<tr>
<td>PS25BEAC,P25</td>
<td>70</td>
<td>1</td>
<td>69,88</td>
<td>0,05</td>
</tr>
<tr>
<td>PS25BEAC,SKY</td>
<td>232</td>
<td>4</td>
<td>58,09</td>
<td>0,17</td>
</tr>
<tr>
<td>PS25BEAC,STD</td>
<td>60</td>
<td>1</td>
<td>60</td>
<td>0,04</td>
</tr>
<tr>
<td>PS25BR</td>
<td>86</td>
<td>1</td>
<td>85,6</td>
<td>0,06</td>
</tr>
<tr>
<td>PS25BTAC</td>
<td>5837</td>
<td>58</td>
<td>100,64</td>
<td>4,30</td>
</tr>
<tr>
<td>PS25BTAC,P25</td>
<td>726</td>
<td>6</td>
<td>121</td>
<td>0,54</td>
</tr>
<tr>
<td>PS25BTAC,P35</td>
<td>74</td>
<td>1</td>
<td>73,79</td>
<td>0,05</td>
</tr>
<tr>
<td>PS25BTAC,SKY</td>
<td>2118</td>
<td>16</td>
<td>132,38</td>
<td>1,56</td>
</tr>
<tr>
<td>PS25BTAC,STD</td>
<td>3641</td>
<td>33</td>
<td>110,35</td>
<td>2,68</td>
</tr>
<tr>
<td>PS42BAAC</td>
<td>9599</td>
<td>72</td>
<td>133,32</td>
<td>7,07</td>
</tr>
<tr>
<td>PS42BAAC,SKY</td>
<td>2763</td>
<td>18</td>
<td>153,51</td>
<td>2,04</td>
</tr>
<tr>
<td>PS42BAAC,STD</td>
<td>2738</td>
<td>17</td>
<td>161,05</td>
<td>2,02</td>
</tr>
<tr>
<td>PS42BAAC,ZDV</td>
<td>1339</td>
<td>8</td>
<td>167,37</td>
<td>0,99</td>
</tr>
<tr>
<td>PS42BBAC</td>
<td>6890</td>
<td>56</td>
<td>123,04</td>
<td>5,08</td>
</tr>
<tr>
<td>PS42BBAC,P35</td>
<td>137</td>
<td>1</td>
<td>137,47</td>
<td>0,10</td>
</tr>
<tr>
<td>PS42BBAC,SKY</td>
<td>2009</td>
<td>14</td>
<td>143,48</td>
<td>1,48</td>
</tr>
<tr>
<td>PS42BBAC,STD</td>
<td>3110</td>
<td>21</td>
<td>148,09</td>
<td>2,29</td>
</tr>
<tr>
<td>PS42BBAC,ZDV</td>
<td>160</td>
<td>1</td>
<td>160</td>
<td>0,12</td>
</tr>
<tr>
<td>PS42BEAC</td>
<td>446</td>
<td>4</td>
<td>111,47</td>
<td>0,33</td>
</tr>
<tr>
<td>PS42BEAC,SKY</td>
<td>461</td>
<td>3</td>
<td>153,63</td>
<td>0,34</td>
</tr>
<tr>
<td>PS42BEAC,STD</td>
<td>538</td>
<td>4</td>
<td>134,46</td>
<td>0,40</td>
</tr>
<tr>
<td>PS42BTAC</td>
<td>3642</td>
<td>30</td>
<td>121,39</td>
<td>2,68</td>
</tr>
<tr>
<td>PS42BTAC,SKY</td>
<td>1481</td>
<td>11</td>
<td>134,68</td>
<td>1,09</td>
</tr>
<tr>
<td>PS42BTAC,STD</td>
<td>1959</td>
<td>14</td>
<td>139,93</td>
<td>1,44</td>
</tr>
<tr>
<td>PS42BTAC,ZDV</td>
<td>143</td>
<td>1</td>
<td>142,74</td>
<td>0,11</td>
</tr>
<tr>
<td>PS92BB</td>
<td>827</td>
<td>6</td>
<td>137,81</td>
<td>0,61</td>
</tr>
<tr>
<td>PS92BB,ZDV</td>
<td>120</td>
<td>1</td>
<td>120,11</td>
<td>0,09</td>
</tr>
<tr>
<td>SKY</td>
<td>2441</td>
<td>28</td>
<td>87,19</td>
<td>1,80</td>
</tr>
<tr>
<td>SKY,P25AT,PS</td>
<td>132</td>
<td>1</td>
<td>131,73</td>
<td>0,10</td>
</tr>
<tr>
<td>SKY,PS25BAAC</td>
<td>1416</td>
<td>12</td>
<td>117,98</td>
<td>1,04</td>
</tr>
<tr>
<td>SKY,PS25BB</td>
<td>430</td>
<td>4</td>
<td>107,42</td>
<td>0,32</td>
</tr>
<tr>
<td>SKY,PS25BBAC</td>
<td>536</td>
<td>4</td>
<td>134,12</td>
<td>0,40</td>
</tr>
</tbody>
</table>
This is an easy understanding list. The letters stands for:

P = Polycarbonate
PS = Sandwich Panel
SKY = Skyplane, ‘high end’ double glass
STDV = normal double glass
ZDV = without roofing material

The letters are followed by a number which gives the thickness of the roofing material. The letters following the digits give information about the colour. Sometimes a second roofing material is used for the same veranda. The surface covered by this second roofing material is much less important than the one covered by the first listed material. For the analysis, only the main material was charged. Of course verandas without roofing material doesn’t exist; ZDV stands for the verandas purchased by the client at the company, but where the glass was bought by the client at another company, so Veranco handles this case as verandas without roofing material.

With these data the ‘roofing material’ analysis can be made. For example, for the 16 mm Polycarbonate (P16) all data beginning with P16 are added up: 957 + 38 + 39 + 2027 + 136 =
The same thing can be done for the 25 mm Polycarbonate, 35 mm Polycarbonate, Sandwich Panel and Double Glazing. For the overall ‘roofing material’ graph, all Sandwich Panels thicknesses are added up together.

For the 25 mm Polycarbonate a deeper analysis was made. After P25, we find the following letters: C, O, B, AT, HS and F, some of which are abbreviations of ‘Clair’ (Clear), ‘Opale’ (Opal), ‘Blanc’ (White), ‘Athermic’. HS and F are not used anymore since 2005. The low amount of P25HS verandas is caused by a veranda bought in 2004 and made in 2005.

A deep analysis was also made for the Sandwich Panel. The analysis is analogous to this for the 25 mm Polycarbonate.

### 2.1.3 Colour

To make the ‘colour’ analysis a third and last list was used:

<table>
<thead>
<tr>
<th>Colour</th>
<th>Points</th>
<th>Number</th>
<th>Mean</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>BICOLOR</td>
<td>6169</td>
<td>40</td>
<td>154,22</td>
<td>3,17</td>
</tr>
<tr>
<td>DUO</td>
<td>3929</td>
<td>35</td>
<td>112,27</td>
<td>2,02</td>
</tr>
<tr>
<td>DUO'EQ</td>
<td>103</td>
<td>1</td>
<td>102,61</td>
<td>0,05</td>
</tr>
<tr>
<td>DUOAS.</td>
<td>90</td>
<td>1</td>
<td>90</td>
<td>0,05</td>
</tr>
<tr>
<td>DUOETE</td>
<td>50</td>
<td>1</td>
<td>49,53</td>
<td>0,03</td>
</tr>
<tr>
<td>DUOGTO</td>
<td>152</td>
<td>1</td>
<td>151,77</td>
<td>0,08</td>
</tr>
<tr>
<td>LB</td>
<td>44198</td>
<td>562</td>
<td>78,64</td>
<td>22,72</td>
</tr>
<tr>
<td>LC</td>
<td>49784</td>
<td>487</td>
<td>102,23</td>
<td>25,59</td>
</tr>
<tr>
<td>LC?</td>
<td>411</td>
<td>3</td>
<td>137,1</td>
<td>0,21</td>
</tr>
<tr>
<td>LM</td>
<td>17001</td>
<td>245</td>
<td>69,39</td>
<td>8,74</td>
</tr>
<tr>
<td>LMAT</td>
<td>21593</td>
<td>206</td>
<td>104,82</td>
<td>11,10</td>
</tr>
<tr>
<td>LP</td>
<td>50916</td>
<td>620</td>
<td>82,12</td>
<td>26,17</td>
</tr>
<tr>
<td>Z</td>
<td>27</td>
<td>3</td>
<td>9,16</td>
<td>0,01</td>
</tr>
</tbody>
</table>

This is again an easy to understand list. Bicolor and Duo stands for verandas made of two different colours. Duo means a mix of profiles of two different colours. A Bicolor veranda has profiles with two different colours. LB, LM and LP are the three most used colours, also the colours the company has continuously in stock: white, chestnut and beige. LC stands for the 176 lacquered colours and LMAT for the nine lacquered matt colours.
2.2 **P-Q analysis**

A product – quantity analysis gives the ability to gain an inside into the importance of every product or sub product and to see which are the ‘fast movers’ and ‘slow movers’. The P-Q analysis is important to know on which products we have to focus for our study. The analysis was made over a period of five years, from 2001 until 2005, to get a better view of the evolution of the sales.
2.2.1 Roof profile types

The first P-Q graph gives a view on the different products.

The used abbreviations for each kind of roof profile are the following:

VA  = Veranda Ariane  P-PL  = Perlanda – Pergola Wooden Verandas
VC  = Veranda Confort   MS  = Menuiserie (windows and doors)
VP  = Veranda Prestige+

<table>
<thead>
<tr>
<th>Type</th>
<th>2001</th>
<th>2002</th>
<th>2003</th>
<th>2004</th>
<th>2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>VA</td>
<td>16390</td>
<td>14489</td>
<td>13986</td>
<td>11719</td>
<td>8043</td>
</tr>
<tr>
<td>VC</td>
<td>55032</td>
<td>59578</td>
<td>70683</td>
<td>64307</td>
<td>61852</td>
</tr>
<tr>
<td>VP</td>
<td>67015</td>
<td>63854</td>
<td>76543</td>
<td>93646</td>
<td>112383</td>
</tr>
<tr>
<td>P-PL</td>
<td>16258</td>
<td>15462</td>
<td>10494</td>
<td>7355</td>
<td>5986</td>
</tr>
<tr>
<td>MS</td>
<td>7330</td>
<td>6919</td>
<td>7438</td>
<td>7298</td>
<td>5570</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>2001</th>
<th>2002</th>
<th>2003</th>
<th>2004</th>
<th>2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>VA</td>
<td>162025</td>
<td>160303</td>
<td>179145</td>
<td>184325</td>
<td>193835</td>
</tr>
<tr>
<td>VC</td>
<td>160303</td>
<td>15462</td>
<td>76543</td>
<td>93646</td>
<td>112383</td>
</tr>
<tr>
<td>VP</td>
<td>162025</td>
<td>160303</td>
<td>179145</td>
<td>184325</td>
<td>193835</td>
</tr>
<tr>
<td>P-PL</td>
<td>162025</td>
<td>160303</td>
<td>179145</td>
<td>184325</td>
<td>193835</td>
</tr>
<tr>
<td>MS</td>
<td>160303</td>
<td>15462</td>
<td>76543</td>
<td>93646</td>
<td>112383</td>
</tr>
</tbody>
</table>

Table 2-4  P-Q analysis of the Roof Profile Types

This is a very interesting graph that tells us a lot of things. The Confort (VC) and Prestige+ (VP) are by far the two most important kinds. The Ariane is already making less than 5 % of their market, and has been halved during the passed five years. The same thing can be said for the wooden verandas. The Menuiserie (MS) has always been low, a normal thing as the company is specialised in complete verandas and not in windows or doors. Because of this, we will only study the Confort (VC) and Prestige+ (VP) product lines.
2.2.2 Thermal break

In following graph we see the evolution of the thermal break. As already mentioned, the Confort line is partially provided of profiles with thermal breaks and the Prestige+ verandas completely. The used abbreviations are:

- PLP = Perlanda Prestige+
- PLC – PG = Perlanda Confort –Pergola
- MC = Menuiserie Confort
- MP = Menuiserie Prestige+

### Table 2-5  P-Q analysis of the Thermal Break

<table>
<thead>
<tr>
<th></th>
<th>2001</th>
<th></th>
<th>2002</th>
<th></th>
<th>2003</th>
<th></th>
<th>2004</th>
<th></th>
<th>2005</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>points</td>
<td>%</td>
<td>points</td>
<td>%</td>
<td>points</td>
<td>%</td>
<td>points</td>
<td>%</td>
<td>points</td>
<td>%</td>
</tr>
<tr>
<td>VA</td>
<td>16390</td>
<td>10,12</td>
<td>14489</td>
<td>9,04</td>
<td>13986</td>
<td>7,81</td>
<td>11719</td>
<td>6,36</td>
<td>8043</td>
<td>4,15</td>
</tr>
<tr>
<td>VC</td>
<td>55032</td>
<td>33,97</td>
<td>59578</td>
<td>37,17</td>
<td>70683</td>
<td>39,46</td>
<td>64307</td>
<td>34,89</td>
<td>61852</td>
<td>31,91</td>
</tr>
<tr>
<td>VP</td>
<td>67015</td>
<td>41,36</td>
<td>63854</td>
<td>39,83</td>
<td>76543</td>
<td>42,73</td>
<td>93646</td>
<td>50,80</td>
<td>112383</td>
<td>57,98</td>
</tr>
<tr>
<td>PLP</td>
<td>4549</td>
<td>2,81</td>
<td>4739</td>
<td>2,96</td>
<td>2077</td>
<td>1,16</td>
<td>2903</td>
<td>1,57</td>
<td>2049</td>
<td>1,06</td>
</tr>
<tr>
<td>PLC - PG</td>
<td>11708</td>
<td>7,23</td>
<td>10723</td>
<td>6,69</td>
<td>8417</td>
<td>4,70</td>
<td>4453</td>
<td>2,42</td>
<td>3937</td>
<td>2,03</td>
</tr>
<tr>
<td>MC</td>
<td>4592</td>
<td>2,83</td>
<td>4573</td>
<td>2,85</td>
<td>4338</td>
<td>2,42</td>
<td>3491</td>
<td>1,89</td>
<td>2196</td>
<td>1,13</td>
</tr>
<tr>
<td>MP</td>
<td>2737</td>
<td>1,69</td>
<td>2347</td>
<td>1,46</td>
<td>3099</td>
<td>1,73</td>
<td>3807</td>
<td>2,07</td>
<td>3375</td>
<td>1,74</td>
</tr>
<tr>
<td>TOT</td>
<td>162025</td>
<td>100</td>
<td>160303</td>
<td>100</td>
<td>179145</td>
<td>100</td>
<td>184325</td>
<td>100</td>
<td>193835</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2001</th>
<th></th>
<th>2002</th>
<th></th>
<th>2003</th>
<th></th>
<th>2004</th>
<th></th>
<th>2005</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>with</td>
<td></td>
<td>without</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VA</td>
<td>74302</td>
<td>45,86</td>
<td>87722</td>
<td>54,14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VC</td>
<td>59794</td>
<td>44,25</td>
<td>89363</td>
<td>55,75</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VP</td>
<td>70683</td>
<td>39,46</td>
<td>97425</td>
<td>55,38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLP</td>
<td>4549</td>
<td>2,81</td>
<td>81720</td>
<td>45,62</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLC - PG</td>
<td>11708</td>
<td>7,23</td>
<td>100356</td>
<td>54,45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MC</td>
<td>4592</td>
<td>2,83</td>
<td>117808</td>
<td>60,78</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MP</td>
<td>2737</td>
<td>1,69</td>
<td>83969</td>
<td>45,55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We notice the upwards trend of the Thermal Break, an expected result as the Prestige+ line gained market compared to the Confort line (see ‘roof profile’ graph).
2.2.3 Standard shapes

In this section we can see which shapes are popular and which are less.

### Standard Shapes

<table>
<thead>
<tr>
<th>Shape</th>
<th>2001 points</th>
<th>2002 points</th>
<th>2003 points</th>
<th>2004 points</th>
<th>2005 points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agate</td>
<td>13932</td>
<td>14523</td>
<td>18869</td>
<td>19778</td>
<td>21649</td>
</tr>
<tr>
<td>Topaze</td>
<td>9837</td>
<td>8050</td>
<td>10260</td>
<td>11351</td>
<td>12838</td>
</tr>
<tr>
<td>Emeraude</td>
<td>27137</td>
<td>30043</td>
<td>36353</td>
<td>36474</td>
<td>38589</td>
</tr>
<tr>
<td>Diamant</td>
<td>6423</td>
<td>6116</td>
<td>9251</td>
<td>9551</td>
<td>8642</td>
</tr>
<tr>
<td>Victorienne</td>
<td>5255</td>
<td>4824</td>
<td>5347</td>
<td>3667</td>
<td>3866</td>
</tr>
<tr>
<td>Rubis</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5339</td>
<td>7245</td>
</tr>
<tr>
<td>Grenat</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3866</td>
</tr>
<tr>
<td><strong>TOT</strong></td>
<td><strong>62584</strong></td>
<td><strong>63556</strong></td>
<td><strong>80080</strong></td>
<td><strong>88273</strong></td>
<td><strong>94816</strong></td>
</tr>
</tbody>
</table>

Table 2-6  P-Q analysis of the Standard Shapes

The Emeraude has always been by far the most popular one, followed by the Agata and Topaze. Rubis and Grenat are new shapes, introduced in 2004. The Rubis shape clearly did not miss his introduction, with 6% in 2004 and almost 8% in 2005. It already passed the Victorienne and will probably pass the Diamant in the future.

![Standard Shapes](image)

Figure 2-3  P-Q graph of the Standard Shapes
2.2.4 Roofing materials

Following graph gives the evolution of the used roofing materials.

The used abbreviations are:

P16_ = Polycarbonate 16 mm  
P25_ = Polycarbonate 25 mm  
P35RE = Polycarbonate 35 mm Reflex  
PS_ = Sandwich Panel

<table>
<thead>
<tr>
<th>Roofing Material</th>
<th>2001</th>
<th>2002</th>
<th>2003</th>
<th>2004</th>
<th>2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>P16_</td>
<td>6403</td>
<td>3535</td>
<td>4534</td>
<td>4215</td>
<td>3197</td>
</tr>
<tr>
<td>P25_</td>
<td>79946</td>
<td>71079</td>
<td>78809</td>
<td>68373</td>
<td>54000</td>
</tr>
<tr>
<td>P35RE</td>
<td>3250</td>
<td>3353</td>
<td>4066</td>
<td>4305</td>
<td>3263</td>
</tr>
<tr>
<td>PS_</td>
<td>48996</td>
<td>44315</td>
<td>60744</td>
<td>79789</td>
<td>94513</td>
</tr>
<tr>
<td>Double Glazing</td>
<td>13592</td>
<td>14130</td>
<td>15742</td>
<td>13069</td>
<td>17352</td>
</tr>
</tbody>
</table>

Table 2-7  P-Q analysis of the Roofing Materials

The roofing material graph is interesting as we observe an important evolution from the Polycarbonate to the Sandwich Panel. The most used Polycarbonate has always been the 25 mm, but it went down from 53 % to 31% relative to the total sales. At the same time, the Sandwich Panel increased from 32% to 55%. The market share of double glazing is relatively stable and increasing with the total sales.
### 2.2.4.1 Polycarbonate 25 mm

To understand the previous graph a bit better, we can make a deeper analyse of the Polycarbonate 25 mm and the Sandwich Panel.

There are 4 different types of Polycarbonate 25 mm, all with five partitions: ‘Clair’ (Clear), ‘Opale’ (Opal), ‘Blanc’ (White), ‘Athermic’. The last one is a special polycarbonate with a good light transmission and interesting solar factor. So it let pass a lot of light without heating up the room a lot.

<table>
<thead>
<tr>
<th>P25</th>
<th>2001</th>
<th>2002</th>
<th>2003</th>
<th>2004</th>
<th>2005</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>points</td>
<td>%</td>
<td>points</td>
<td>%</td>
<td>points</td>
</tr>
<tr>
<td>P25C</td>
<td>7290</td>
<td>9.12</td>
<td>6600</td>
<td>8.63</td>
<td>6038</td>
</tr>
<tr>
<td>P25B</td>
<td>16775</td>
<td>20.98</td>
<td>19882</td>
<td>25.98</td>
<td>22352</td>
</tr>
<tr>
<td>P25AT</td>
<td>43968</td>
<td>55.00</td>
<td>41584</td>
<td>54.34</td>
<td>42833</td>
</tr>
<tr>
<td>P25F</td>
<td>1393</td>
<td>1.74</td>
<td>434</td>
<td>0.57</td>
<td>64</td>
</tr>
<tr>
<td>P25HS</td>
<td>229</td>
<td>0.29</td>
<td>867</td>
<td>1.27</td>
<td>21</td>
</tr>
</tbody>
</table>

**TOT** | 79946 | 100    | 76520  | 100    | 78809  | 100    | 68373  | 100    | 54000  | 100    |

Table 2-8  P-Q analysis of the Polycarbonate 25mm

The Athermic is by far the most popular one. When we look at the absolute numbers, we perceive that from 2003 to 2004 and from 2004 to 2005 all the different types decreased. The white one decreased a lot which can be explained by the rice of the Sandwich Panel which are always white.
2.2.4.2 Sandwich panel

In this graph all the different Sandwich Panels are projected. The used abbreviations are:

PS25BB  = Sandwich Panel 25 mm blanc/ blanc
PS25BA  = Sandwich Panel 25 mm blanc/ ardoise
PS25BE  = Sandwich Panel 25 mm blanc/ esterel moucheté
PS25BT  = Sandwich Panel 25 mm blanc/ tuile
PS42    = Sandwich Panel 42 mm
PS92    = Sandwich Panel 92 mm

<table>
<thead>
<tr>
<th>PS</th>
<th>2001 points</th>
<th>%</th>
<th>2002 points</th>
<th>%</th>
<th>2003 points</th>
<th>%</th>
<th>2004 points</th>
<th>%</th>
<th>2005 points</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS25BB</td>
<td>29623</td>
<td>60.46</td>
<td>24939</td>
<td>56.28</td>
<td>27871</td>
<td>45.88</td>
<td>27714</td>
<td>34.73</td>
<td>29799</td>
<td>31.53</td>
</tr>
<tr>
<td>PS25BE</td>
<td>6236</td>
<td>12.73</td>
<td>6189</td>
<td>13.96</td>
<td>6021</td>
<td>9.91</td>
<td>3241</td>
<td>4.06</td>
<td>3117</td>
<td>3.30</td>
</tr>
<tr>
<td>PS25BT</td>
<td>6768</td>
<td>8.48</td>
<td>12396</td>
<td>13.12</td>
<td>29569</td>
<td>37.06</td>
<td>28745</td>
<td>30.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PS42</td>
<td>5000</td>
<td>10.21</td>
<td>4711</td>
<td>10.63</td>
<td>13861</td>
<td>22.82</td>
<td>29569</td>
<td>37.06</td>
<td>28745</td>
<td>30.41</td>
</tr>
<tr>
<td>PS92</td>
<td>5000</td>
<td>10.21</td>
<td>4711</td>
<td>10.63</td>
<td>13861</td>
<td>22.82</td>
<td>29569</td>
<td>37.06</td>
<td>28745</td>
<td>30.41</td>
</tr>
<tr>
<td>TOT</td>
<td>48996</td>
<td>100</td>
<td>44315</td>
<td>100</td>
<td>60744</td>
<td>100</td>
<td>79789</td>
<td>100</td>
<td>94513</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 2-9  P-Q analysis of the Sandwich Panel

Figure 2-6  P-Q graph of the Sandwich panel

We notice a huge increase of the 42 mm Sandwich Panel, which is even more spectacular in absolute numbers.
2.2.5 Colours

This last P-Q graph gives an idea of the colours used.

LB = ‘Laque blanc neige’ white
LM = ‘Laque marron normand’ chestnut
LP = ‘Laque beige ton pierre’ beige
LC = lacquered colours
LMAT = lacquered matt

<table>
<thead>
<tr>
<th>Colour</th>
<th>2001 points</th>
<th>2002 points</th>
<th>2003 points</th>
<th>2004 points</th>
<th>2005 points</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB</td>
<td>44027</td>
<td>43226</td>
<td>45298</td>
<td>44822</td>
<td>44198</td>
</tr>
<tr>
<td>LM</td>
<td>27889</td>
<td>25932</td>
<td>23726</td>
<td>16896</td>
<td>17001</td>
</tr>
<tr>
<td>LP</td>
<td>45488</td>
<td>40818</td>
<td>52790</td>
<td>50674</td>
<td>50916</td>
</tr>
<tr>
<td>LC</td>
<td>38314</td>
<td>40513</td>
<td>43526</td>
<td>50261</td>
<td>50195</td>
</tr>
<tr>
<td>LMAT</td>
<td>5107</td>
<td>7730</td>
<td>9784</td>
<td>12273</td>
<td>21593</td>
</tr>
<tr>
<td>BICOLOR</td>
<td>0</td>
<td>2469</td>
<td>4353</td>
<td>10312</td>
<td>6169</td>
</tr>
<tr>
<td>DUO</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4078</td>
<td>4323</td>
</tr>
</tbody>
</table>

| Total    | 160826      | 160688      | 179476      | 189315      | 194395      |

Table 2-10 P-Q analysis of the Colours

White, chestnut and beige are still the most used colours, but the six promoted lacquered colours ‘LC6’ are coming close. So, until now the three main colours stays in stock, but in the future they will have to look at the important LC6 colours. We notice that the LC in the graph is the sum of all the LC colours.

Figure 2-7 P-Q graph of the Colours
2.3 Elaborate example of a veranda project and BOM

This section gives an elaborate example of a veranda, like it is drawn in the engineering firm accompanied with the BOM (Bill Of Material) of the roof profiles and accessories. Later in Chapter 4 this veranda project will be used to illustrate the material waste problem.

The first page gives a general overview of the veranda. On the top we see the name of the client, Loiseau in this case, and the file number: 051481. The following line gives information about the clients’ company name and the date in which the order was received by Veranco. Subsequently we see: the veranda type, VC (Veranda Confort); the colour (lacquered white, RAL 9010) and the type of double glazing (4 – 16 – 4). The last line gives information about: the roof material P25AT (25 mm Polycarbonate Athermic); the side triangle Polyplus (a sort of double glass) and the person who drew the veranda. Below the head text, two plans are given with all necessary dimensions and details as the placement of the spots and the type of windows and doors, fixed (F) or sliding (C). Above the first plan, the height of the veranda according to different points is given. From the plans we can deduce that it is an Agate Veranda, which is the less complicated veranda with a rectangular shape.

On the top right of the second page, the veranda type is written down like was found in the raw data list: VCA (Veranda Agate Confort). This page gives detail plans of the front and the side of the veranda, with the side triangles. As we have seen in Section 1.4, these side triangles are required because it is an Agate Veranda. In this specific case we can see that the client is asking an extra C2CC (on the bottom right of the little drawings), this is a frame with two windows, which both are sliding one. This has no relation with this veranda, but it can be a piece of a previous order that was broken or wrong ordered. It illustrates the possibility to send additional parts of previous orders. The client puts this little order together with a new veranda order, so they will be packed together and the little order will not get lost for sure. Below the names of the file, client and company we observe the interior colour and exterior colour. In this case they are the same, but as we have seen, it is possible to have an exterior colour different from the interior colour, and the veranda is considered ‘Bicolor’. On the bottom information is given about the need of pre-assembling and the total number of window and door frames as a check. This is an easy rectangular shape, so no pre-montage is needed.
The following three pages give the profile BOM, extracted from the drawing and the veranda type. This profile BOM will be further used in the material waste optimization problem. For each piece that has to be cut, we see a little drawing, the profile reference and ID, a description, the colour, the length and the horizontal and vertical angle of the two edges.

The next three pages give the accessory BOM, where the quantity of the elements is given instead of the length.

The last page is the roofing material BOM and gives information about the type, thickness and length of the roofing material and the side triangles. Below, we can see if they have to be packed in a wooden box or not.
Rapport de fabrication

DOSSIER : 051481
REFERENTIE : LOISEAU
CLIENT : ALU FERMETURES

Couleur Exterieur 9010  Interieur 9010

REMARQUES

PREMONTAGE : NON  NOMBRE DE CHASSIS : 8

EXTRA C2CC !!!!
### Profils

**ALU FERMETURES**

#### Cheneau

<table>
<thead>
<tr>
<th>Element</th>
<th>ID</th>
<th>Reference</th>
<th>Description</th>
<th>Couleur</th>
<th>Long.</th>
<th>H1</th>
<th>V1</th>
<th>H2</th>
<th>V2</th>
</tr>
</thead>
<tbody>
<tr>
<td>18_1</td>
<td></td>
<td>VER 06/14</td>
<td>Cheneau</td>
<td>9010</td>
<td></td>
<td>3252</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24_1</td>
<td></td>
<td>VER 15/25</td>
<td>Profil d’embout polycarbonate</td>
<td>9010</td>
<td></td>
<td>1043</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25_1</td>
<td></td>
<td>VER 15/25</td>
<td>Profil d’embout polycarbonate</td>
<td>9010</td>
<td></td>
<td>1043</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26_1</td>
<td></td>
<td>VER 15/25</td>
<td>Profil d’embout polycarbonate</td>
<td>9010</td>
<td></td>
<td>1043</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29_1</td>
<td></td>
<td>VER 59</td>
<td>Rehausse</td>
<td>9010</td>
<td></td>
<td>3252</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Chevrons

<table>
<thead>
<tr>
<th>Element</th>
<th>ID</th>
<th>Reference</th>
<th>Description</th>
<th>Couleur</th>
<th>Long.</th>
<th>H1</th>
<th>V1</th>
<th>H2</th>
<th>V2</th>
</tr>
</thead>
<tbody>
<tr>
<td>19_1</td>
<td></td>
<td>VER 01/25A</td>
<td>Chevron pour capse pvc</td>
<td>9010</td>
<td></td>
<td>1924</td>
<td>8.45</td>
<td>8.45</td>
<td></td>
</tr>
<tr>
<td>20_1</td>
<td></td>
<td>VER 01/25A</td>
<td>Chevron pour capse pvc</td>
<td>9010</td>
<td></td>
<td>1924</td>
<td>8.45</td>
<td>8.45</td>
<td></td>
</tr>
<tr>
<td>21_1</td>
<td></td>
<td>VER 01/25A</td>
<td>Chevron pour capse pvc</td>
<td>9010</td>
<td></td>
<td>1924</td>
<td>8.45</td>
<td>8.45</td>
<td></td>
</tr>
<tr>
<td>31_1</td>
<td></td>
<td>VER 01/25A</td>
<td>Chevron pour capse pvc</td>
<td>9010</td>
<td></td>
<td>1924</td>
<td>8.45</td>
<td>8.45</td>
<td></td>
</tr>
<tr>
<td>14_1</td>
<td></td>
<td>VER 02/25</td>
<td>Chevron latéral</td>
<td>9010</td>
<td></td>
<td>1924</td>
<td>8.45</td>
<td>8.45</td>
<td></td>
</tr>
<tr>
<td>8_1</td>
<td></td>
<td>PVC PROFIEL 01</td>
<td>PVC Clips</td>
<td></td>
<td></td>
<td>1952</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16_1</td>
<td></td>
<td>PVC PROFIEL 01</td>
<td>PVC Clips</td>
<td></td>
<td></td>
<td>1952</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Data Extraction and P-Q Analysis

<table>
<thead>
<tr>
<th>Element ID</th>
<th>Reference</th>
<th>Description</th>
<th>Colour</th>
<th>Long.</th>
<th>H1</th>
<th>V1</th>
<th>H2</th>
<th>V2</th>
</tr>
</thead>
<tbody>
<tr>
<td>27_1</td>
<td>PVCPROFIEL 01</td>
<td>PVC Clips</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28_1</td>
<td>PVCPROFIEL 01</td>
<td>PVC Clips</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7_1</td>
<td>E 02/25</td>
<td>Parclose chevron lateral</td>
<td>9010</td>
<td>1948</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15_1</td>
<td>E 02/25</td>
<td>Parclose chevron lateral</td>
<td>9010</td>
<td>1948</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31_1</td>
<td>TRAVERSE 01</td>
<td>Tube 30x30</td>
<td>9010</td>
<td>1030</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32_1</td>
<td>TRAVERSE 01</td>
<td>Tube 30x30</td>
<td>9010</td>
<td>1030</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33_1</td>
<td>TRAVERSE 01</td>
<td>Tube 30x30</td>
<td>9010</td>
<td>1030</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Faitières

<table>
<thead>
<tr>
<th>Element ID</th>
<th>Reference</th>
<th>Description</th>
<th>Colour</th>
<th>Long.</th>
<th>H1</th>
<th>V1</th>
<th>H2</th>
<th>V2</th>
</tr>
</thead>
<tbody>
<tr>
<td>17_1</td>
<td>VER 05/25C</td>
<td>Faïbère</td>
<td>9010</td>
<td>3250</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21_1</td>
<td>VER 06</td>
<td>Support de plaque&lt;br&gt;Links 26.1 mm uitwerken aan zijdeklijger&lt;br&gt;• RUBBER 26</td>
<td>9010</td>
<td>1022</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22_1</td>
<td>VER 06</td>
<td>Support de plaque&lt;br&gt;• RUBBER 26</td>
<td>9010</td>
<td>1010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23_1</td>
<td>VER 06</td>
<td>Support de plaque&lt;br&gt;Rechts 26.1 mm uitwerken aan zijdeklijger&lt;br&gt;• RUBBER 26</td>
<td>9010</td>
<td>1022</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30_1</td>
<td>SOLIN 01</td>
<td>Solin</td>
<td>9010</td>
<td>3250</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Poteaux

<table>
<thead>
<tr>
<th>Element ID</th>
<th>Reference</th>
<th>Description</th>
<th>Colour</th>
<th>Long.</th>
<th>H1</th>
<th>V1</th>
<th>H2</th>
<th>V2</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>VER 73</td>
<td>Tube 100x130</td>
<td>9010</td>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>VER 73</td>
<td>Tube 100x130</td>
<td>9010</td>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AVE DECÖUPE DE COTE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Chapter 2  Data Extraction and P-Q Analysis

<table>
<thead>
<tr>
<th>Element ID</th>
<th>Reference</th>
<th>Description</th>
<th>Couleur</th>
<th>Long</th>
<th>H1</th>
<th>V1</th>
<th>H2</th>
<th>V2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1_1</td>
<td>VER 72</td>
<td>Profile de Rive</td>
<td>9010</td>
<td>1868</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ Rubber 08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2_1</td>
<td>VER 72</td>
<td>Profile de Rive</td>
<td>9010</td>
<td>275</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ Rubber 08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3_1</td>
<td>VER 72</td>
<td>Profile de Rive</td>
<td>9010</td>
<td>1868</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ Rubber 08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4_1</td>
<td>VER 72</td>
<td>Profile de Rive</td>
<td>9010</td>
<td>275</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ Rubber 08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5_1</td>
<td>VER 11/10</td>
<td>Pareclose (poly 25mm)</td>
<td>9010</td>
<td>1878</td>
<td>8,45</td>
<td>8,45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6_1</td>
<td>VER 11/10</td>
<td>Pareclose (poly 25mm)</td>
<td>9010</td>
<td>1868</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7_1</td>
<td>VER 11/10</td>
<td>Pareclose (poly 25mm)</td>
<td>9010</td>
<td>275</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8_1</td>
<td>VER 11/10</td>
<td>Pareclose (poly 25mm)</td>
<td>9010</td>
<td>1878</td>
<td>8,45</td>
<td>8,45</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Profile LUTZ

<table>
<thead>
<tr>
<th>Element ID</th>
<th>Reference</th>
<th>Description</th>
<th>Couleur</th>
<th>Long</th>
<th>H1</th>
<th>V1</th>
<th>H2</th>
<th>V2</th>
</tr>
</thead>
<tbody>
<tr>
<td>34_1</td>
<td>P 30x2</td>
<td>Plat 30x2</td>
<td>9010</td>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35_1</td>
<td>P 30x2</td>
<td>Plat 30x2</td>
<td>9010</td>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36_1</td>
<td>P 30x2</td>
<td>Plat 30x2</td>
<td>9010</td>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37_1</td>
<td>P 30x2</td>
<td>Plat 30x2</td>
<td>9010</td>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Accessoires

<table>
<thead>
<tr>
<th>Element</th>
<th>Reference</th>
<th>Description</th>
<th>Couleur</th>
<th>Quant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPOT 04</td>
<td>Spot ronde Ø47</td>
<td></td>
<td>9010</td>
<td>3</td>
</tr>
</tbody>
</table>

## Divers

<table>
<thead>
<tr>
<th>Element</th>
<th>Reference</th>
<th>Description</th>
<th>Couleur</th>
<th>Quant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETOUCHEUR</td>
<td>Retoucheur</td>
<td></td>
<td>9010</td>
<td>1</td>
</tr>
</tbody>
</table>

## Fermetures

<table>
<thead>
<tr>
<th>Element</th>
<th>Reference</th>
<th>Description</th>
<th>Couleur</th>
<th>Quant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOUSB 16</td>
<td>Bande etafosm - Ø 15 mm</td>
<td></td>
<td></td>
<td>13,1</td>
</tr>
<tr>
<td>PARABOND WT</td>
<td>Parabond blanc</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>SIL TRANS</td>
<td>Silicones - transparent</td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>MOUSB 10</td>
<td>Bande etafosm - Ø 10 mm</td>
<td></td>
<td></td>
<td>7,1</td>
</tr>
<tr>
<td>RUBBER 03</td>
<td>Joint faîtière</td>
<td></td>
<td></td>
<td>3,4</td>
</tr>
<tr>
<td>RUBBER 22</td>
<td>Joint de rive nr° 2 (2mm)</td>
<td></td>
<td></td>
<td>8,4</td>
</tr>
<tr>
<td>RUBBER 27</td>
<td>Joint cheneau (14 mm)</td>
<td></td>
<td></td>
<td>3,4</td>
</tr>
<tr>
<td>RUBBER 28</td>
<td>Joint cheneau (52 mm)</td>
<td></td>
<td></td>
<td>3,4</td>
</tr>
<tr>
<td>SILICONEN</td>
<td>Silicones</td>
<td></td>
<td>9010</td>
<td>9</td>
</tr>
</tbody>
</table>

## Pièces des fixations

<table>
<thead>
<tr>
<th>Element</th>
<th>Reference</th>
<th>Description</th>
<th>Couleur</th>
<th>Quant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEVSTK 01</td>
<td>Tasseau</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>BEVSTK 03</td>
<td>Tasseau gauche</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
### Plaques d'embout

<table>
<thead>
<tr>
<th>Element</th>
<th>Reference</th>
<th>Description</th>
<th>Couleur</th>
<th>Quant.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PLAKPR 06</td>
<td>Plaque d'embout fâtière Ver 05/25</td>
<td>9010</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>PLAKPR 08</td>
<td>Plaque d'embout de panne (16 et 26 mm)</td>
<td>9010</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>PLAKPR 16</td>
<td>Plaque d'embout pour cheneau VER 66/14 + VER59</td>
<td>9010</td>
<td>2</td>
</tr>
</tbody>
</table>

### Afwatering

<table>
<thead>
<tr>
<th>Element</th>
<th>Reference</th>
<th>Description</th>
<th>Couleur</th>
<th>Quant.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PVC UITLOOP</td>
<td>Crepine pvc Ø80 Lichtgrijs</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>PVC BOCHT 90/67</td>
<td>Coude pvc Ø80 LICHTGRIJS</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

### Fermetures

<table>
<thead>
<tr>
<th>Element</th>
<th>Reference</th>
<th>Description</th>
<th>Couleur</th>
<th>Quant.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SIL TRANS</td>
<td>Silicones - transparent</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

### Menuiserie

<table>
<thead>
<tr>
<th>Element</th>
<th>Reference</th>
<th>Description</th>
<th>Couleur</th>
<th>Quant.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ANTIDIEFSTALBLOK</td>
<td>Pièce anti-vol S840</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>B 19x19x38</td>
<td>Buis 19x19x38</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>GACHE 01 - 640</td>
<td>Gâche 01 (Confort)</td>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>

### Vis

<table>
<thead>
<tr>
<th>Element</th>
<th>Reference</th>
<th>Description</th>
<th>Couleur</th>
<th>Quant.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PLUG 6</td>
<td>Cheville 6 Mm</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>PLUG 8</td>
<td>Cheville 8 Mm</td>
<td></td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>PVC TOP</td>
<td>Cache PVC</td>
<td>9010</td>
<td>86</td>
</tr>
<tr>
<td>Material Code</td>
<td>Description</td>
<td>Quantity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>---------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPLSCHR/6X70</td>
<td>Vis parker tête fraisée 6 x 70</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STELSCHR/KRT/6X</td>
<td>Steischroef Met Kraterind M6x10</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VJUS/C/3.5X19</td>
<td>Vis Parker Tête Ronde 3.5 x 19</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VJUS/C/4.8X13</td>
<td>Vis Parker Tête Ronde 4.8 x 13</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VJUS/C/4.8X19</td>
<td>Vis Parker Tête Ronde 4.9 x 19</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VJUS/C/4.8X25</td>
<td>Vis Parker Tête Ronde 4.8 x 25</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VJUS/C/4.8X38</td>
<td>Vis Parker Tête Ronde 4.8 x 38</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VUS/H6X50</td>
<td>Vis Boul Tête Fraisée 6 x 50</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VUS/S/3.5X13</td>
<td>Vis Parker Tête Fraisée 3.5 x 13</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VUS/S/3.5X19</td>
<td>Vis Parker Tête Fraisée 3.5 x 19</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VUS/S/3.5X32</td>
<td>Vis Parker Tête Fraisée 3.5 x 32</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VUS/S/4.8X13</td>
<td>Vis Parker Tête Fraisée 4.8 x 13</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VUS/S/4.8X19</td>
<td>Vis Parker Tête Fraisée 4.8 x 19</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VUS/S/4.8X26</td>
<td>Vis Parker Tête Fraisée 4.8 x 26</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VUS/S/4.8X32</td>
<td>Vis Parker Tête Fraisée 4.9 x 32</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VUS/S/4.8X38</td>
<td>Vis Parker Tête Fraisée 4.8 x 38</td>
<td>43</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Couverture et triangle de rive

<table>
<thead>
<tr>
<th>Élément</th>
<th>REF</th>
<th>Description</th>
<th>Numéro A.B.C</th>
<th>Quant</th>
<th>Largeur</th>
<th>Longueur</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS25AT</td>
<td></td>
<td>PLATEN OP JUISTE MATEN ZAGEN</td>
<td>3</td>
<td>1043</td>
<td>1940</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>TOTAL: 3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DV</td>
<td></td>
<td>POLY SUPER PLUS K = 1.4</td>
<td>1</td>
<td>1832</td>
<td>302</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REEDS BESTELD BIJ POLYPANE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DV</td>
<td></td>
<td>POLY SUPER PLUS K = 1.4</td>
<td>1</td>
<td>1832</td>
<td>30</td>
<td>302</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REEDS BESTELD BIJ POLYPANE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### CAISSE

<table>
<thead>
<tr>
<th>Élément</th>
<th>Oui / Non</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAISSE / KIST</td>
<td>NON</td>
<td>Kist voor sandwich panelen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Caisse en bois pour panneau sandwich</td>
</tr>
<tr>
<td>CAISSE / KIST</td>
<td>OUI</td>
<td>Kist voor glas panelen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Caisse en bois pour vitrage</td>
</tr>
</tbody>
</table>
Chapter 3

Veranda Production Process

In this chapter we will study the whole production process of a veranda from the moment the order is received towards the transportation to the client. First, some administration is done; thereupon, the veranda is drawn and calculations are made in the engineering firm; and so, the veranda can be fabricated. The fabrication is done in three departments: the roof department, the window department and the accessories department. Finally, the orders are transported to the client.

3.1 General overview

On the next page, the whole production process is overviewed in a flow chart. All the production steps, the three departments and the two subdivisions in the window department are clearly indicated.

3.2 Administration

All the orders enter by fax. When an order is received, it is inputted to the computer. The client has to compute the points of the veranda, using the calculation tables provided by the company. The points give its presupposed price after multiplying by the client factor. Subsequently, a preplanning is made such that a production of 3500 to 4000 points is planned every week.
ROOF DEPARTMENT

Sawing → Preparing → Pre-montage → Roof Material → Packing

WINDOW DEPARTMENT

Sawing → Preparing
   → Preparing Fixed Part → Glazing
   → Preparing Sliding Part → Glazing

SLIDING WINDOWS

FIXED WINDOWS

Preparing → Montage → Glazing

ACCESSORIES DEPARTMENT

Accessories
3.3 Engineering firm

Thereupon the order goes to the responsible of the engineering firm. He spread them out among the engineers. (Figures 3-1 and 3-2) If profiles with a special colour are needed, a colour that is not in stock, the necessary brut profiles are sent to the paint shop and paint in the desired colour. For every profile type, there is a stock profile in brut, and for most of the profile types the three main colours are also in stock. The painting process is outsourced and takes between one and two weeks. Subsequently glass (if needed) and special accessories are ordered. The glass order is controlled by another engineer to avoid wrong orders, because a wrong glass order can cause an important delay. When all the orders are made, the production file is written and the presupposed price is recalculated. This takes approximately 2 to 3 minutes per point. The last step is a final inspection by another engineer, again to reduce mistakes.

Now the file is brought to the overseers of the fabrication unit. The file consists of a package (yellow) for the roof department, two packages for the window department, one (orange) for the fix frames subdivision and one (green) for the sliding frames subdivision and a package for the accessories department. After a little control by the oversee, the file with the different packages are spread out in the factory. The packages for the roof and window department go to the warehouse and the package for the accessories department goes directly to the accessories department.

3.4 Warehouse

Stock profiles (brut profiles and profiles lacquered in the main colours) arrives in containers to the warehouse (Figure 3-3), they are unpacked and stored per profile in the warehouse. In
case of an order with special colours all needed profiles are assembled and put into a container; this is done on the basis of the BOM (Bill Of Material). The container is then sent to the paint shop. After one week, or a little bit longer, if the colour had just been lacquered, so that a new cycle has to be awaited. When the container with lacquered colours arrives, profiles are controlled one by one. (Figure 3-5) Possible mistakes are: not good lacquered, not enough microns, a dent in the profiles, hair on the profile, a wrong number of profiles, wrong colour, … If there is a mistake, profiles are sent back to the paint shop where they are lacquered in a smaller lacquer train, for priority orders.

Now production can be started, as all profiles for the order are available in the company in the desired colour. The profiles are put on one cart for the roof department and one for the window department.

Figure 3-3 Stock profiles arriving
Figure 3-4 Fork-lift places profiles in warehouse
Figure 3-5 Control of lacquer
Figure 3-6 Stock profiles in warehouse
3.5 Roof department

This is the biggest department of the company. The veranda roof, which is by far the most complex part of the veranda, is sawed, prepared, pre-assembled, provided of roof material and packed.

3.5.1 Sawing step

The sawing line chief calls the warehouse to bring the container. First the gutter is sawed on a special sawing machine, which can saw a combination of a vertical and horizontal angle (see Chapter 6). The other profiles are then sawed by a sawyer, selected by the line chief. (Figures 3-9 and 3-10) The sawed profiles are cleaned by a little blower to remove tiny slivers and placed on a cart. Every order is placed on a different cart. (Figure 3-11) When the ordered is completely sawed the cart is moved up to the buffer between the sawing division and the preparing division.
3.5.2 Preparing step

The line chief of the preparing division takes the first order according to the planning and gives it to one of his workers. In this division a lot of little treatments are done: the profiles are milled, bored, spotlight holes are made (Figure 3-13), rubbers are placed (Figure 3-12) and some edges are reinforced. (Figure 3-14)

After preparing the profiles, the cart is placed in a buffer and then there are two possibilities. If pre-montage is needed, the cart will go to the pre-montage division; if not, the cart will immediately go to the packing division.

3.5.3 Pre-montage step

Most of the veranda roofs, all the not rectangular ones are pre-assembled after the preparation step. (Figure 3-15) This is necessary because some little profile cuts can not be calculated. Pre-assembling the roof will of course accelerate the erection of the veranda by the client and make everything more accurate. A lot of little adjustments are done: rafters are trimmed,
bottom of the gutter is adapted to the bottom of the ridge-tile, neoprene is inserted in the gutters, rafters and supporting profiles.

Normally the roof is pre-assembled on little wooden blocks. The veranda is sometimes pre-assembled on its real height, so the roof is supported by posts and the window and door frames. The veranda will be completely assembled in the following three circumstances:

- if a veranda has rising gutters
- if a veranda has roll-down shutter
- if it is a new client

3.5.4 Roof material step

When pre-montage is done, the roof material division is called. The pre-assembled roof is used to take the measurements for the roof material (polycarbonate or sandwich panel), which is cut on the spot. (Figure 3-20) If glass has to be placed on the roof, moulds are made with a sticker containing the file number and sent to the glass manufacture.
The roofs of the verandas that are not pre-assembled are cut in the roof material area. (Figure 3-21) These roofs have rectangular shapes; so, no measurements have to be taken. The cut polycarbonate is cleaned inside by a blower to remove the tiny slivers. Subsequently both sides are covered with tape; on one side a perforated tape is used (Figure 3.22), on the other side not. The cut sandwich panel does not need follow-up treatments.

Once the measurements are taken, the aluminium roof is disassembled and sent to the packing step together with the cut roof material. In some special cases the cart is sent again to the preparing division for some further work that could not have been done before because of unknown dimensions: for example, spotlight holes in rafters.

3.5.5 Packing step of roof department

All roof profiles and roof materials for an order are picked up from the buffer before the packing step. The orders are spread out by the line chief. Normally all profiles are clean and free of tiny slivers, because they are cleaned after every production step. Subsequently a last control is made on the basis of the BOM. The dimensions, number of profiles and type of
profiles are controlled. A plastic film is placed between every profile to protect them. (Figures 3-23 and 3-24) Thereupon all profiles are placed in a carton (Figure 3-25) and the roof material is placed in a wooden box. A sticker with the client’s name, order number, type of veranda is stuck on the box. (Figure 3-26)

3.6 Window department

In this department windows and doors are fabricated. First the profiles are sawed on the desired length. Subsequently the sawed profiles are moved up to one of the two big subdivisions. There is a subdivision for fixed windows and doors and another for sliding windows and doors.
3.6.1 Sawing step

The profiles are brought from the warehouse to the sawyers. In contrast with the sawing step of the roof department, the profiles are sawed by automatic machines. The dimensions are inputted by the engineering firm. Because the profiles are not as big and large as the roof profiles, different orders are placed on the same cart. To keep it organized, a sticker is placed on every profile.

![Figure 3-29 Inputting the dimensions](image1)
![Figure 3-30 Sawing window profiles](image2)

3.6.2 Subdivision for fixed windows and doors

The process consists of a preparation step, a montage step and a glazing step.

- In the preparation step the lock hole is milled; the lock is placed; holes are bored to fix the profile with the post; neoprene is inserted and some other small handlings are made. (Figures 3-31 and 3-32)
- In the montage step the inside and outside window or door frameworks are made, the hinges are placed, …
- In the glazing step the glass is placed into the framework and fixed with glazing beads. Next, rubber is placed between the glass and the glazing beads.
3.6.3 Subdivision for sliding windows and doors

The process consists of two parallel preparation steps (for the sliding and fixed part) and a glazing step. After the profiles are sawed, the sliding part (inside part) and the fixed part (outside part) are prepared.

The preparation of the sliding part holds in: milling the hole for the lock, placing the lock, positioning wheels on the bottom side, putting rubber around the glass, … (Figures 3-33 and 3-34) Subsequently the sliding part goes to the glazing step.

In the preparation step for the outside part inox rays are placed at the bottom; so, the sliding part with wheels can roll over it, holes for the evacuation of water are bored, … This part must obviously not to go to the glazing step, as the sliding part will be placed into it, and goes to the packing area of the roof department.
The glazing step is completely different from the one for fixed windows and doors. No glazing beads are used. The glass is placed on a glazing table, rubber is put all around it and finally the framework is pressed over it. (Figures 3-35 and 3-36)

3.6.4 Packing step of window department

The glazed windows and doors of the same order, coming from the sliding and fix glazing area, are all put together on a wooden pallet board and wrapped with a plastic film.

3.7 Accessories department

In this little department all accessories are picked up from the stock. These accessories are all goods required by the client to build the veranda. Examples of accessories are screws in all different formats, rubbers, silicone in different colours, plastic gutter inlet filter, screw plugs, … (Figure 3-37 and 3-38) The accessories BOM can be consulted in Section 2.3.
3.8 Transport

Different orders for the same client are collected and put together on the same truck. One order consists of:

- the roof profiles, roof material and outside framework of sliding windows and doors packed in the roof department;
- the glazed window and doors placed on a wooden pallet board and wrapped at the packing step of the windows department;
- the accessories packed in a carton.

Transportation to the client is completely outsourced and achieved by two transportation firms.

![Figure 3-39  Loading a truck](image1)
![Figure 3-40  Transport to the client](image2)
Chapter 4

Waste Reduction Problem

The first problem we will discuss in this work is a cutting stock problem. When sawing the seven-meter aluminium stock profiles to the desired length, the company is faced to a huge amount of waste material. The company estimates the waste at 25%. This is an important number, and besides that it is not only a waste of aluminium. These roof profiles are extruded, treated and in most cases provided of a thermal break, and therefore this is expensive. It should be great to design a model that can directly be implemented and run before starting a sawing batch, so that the waste is minimized.

After a little introduction about the cutting stock problem, a review of the literature will be given without going into details on all the different ways to solve it. Next, a classification of cutting and packing problems will be made. Thereupon, we will develop a manner to regroup multiple veranda projects, with a view to minimize the total waste. In the following part of this chapter a first algorithm to solve the cutting stock problem will be designed. We will discuss the specific cutting stock problem for the veranda industry, and thereafter, a second algorithm is developed. At the end of the thesis, a bibliography of the cutting stock problem can be found.

4.1 The Cutting Stock Problem: an introduction

In the cutting stock problem (CSP), we have a supply of pieces (objects) of stock material on one hand and a set of demands for “small” pieces of this material on the other hand. We must satisfy these demands by cutting the required pieces out of the stock pieces. The objective is primarily to minimize the waste that is counted as the lost part of used pieces of stock material. A solution is given by a set of feasible cutting patterns, i.e. assortments of order
pieces that can be cut out of a given piece of stock material, such that their accumulated production of ordered pieces covers the demands.

4.2 Review of the literature

The very first formulation of the CSP was produced by Kantorovich in 1939, although it was translated and published in English [12] only in 1960. Scientific research started about forty five years ago and the number of different formulations and solution methods for the CSP has been fast growing since then. Brooks et al. [1] wrote the first publication in English on the CSP. Gilmore and Gomory’s papers ([6], [7]) were among the first to present efficient solution techniques for the CSP, based on column generation. Vance [15] developed a branch-and-price algorithm using the Dantzig-Wolfe reformulation, and branching directly on variables associated with the choice of cutting patterns. Scheithauer et al. [13] developed a cutting plane algorithm.

Dyckhoff [4] published a typology of cutting and packing problems according to the dimensionality, the kind of assignment, the assortment of large objects and the assortment of small items.

Sweeny & Paternoster [14] provided an exhaustive and well categorized research bibliography of papers on CSP. Interested readers are also referred to the book by Dyckhoff & Finke [5] for a general survey on problems and solution techniques developed before 1990. Other books which give a review of cutting and packing problems are Hinxman [11] and Haessler & Sweeney [10]. Since then, the number of published papers related to the cutting and packing problem has increased strongly every year. Recent publications have been focused on the issue of stabilization and accelerating the column generation, which are beyond the scope of this work.

4.3 Classification of Cutting and Packing Problems

Companies in many industries (paper, glass, aluminium, steel construction, etc.) are faced with the need to purchase standard sizes of material and subsequently cut them to fit their needs. Others need to pack or load small items into large containers for transportation or
storage. While these problems may seem very different on the surface, they are actually closely related. The next section clarifies their commonalties.

The cutting and packing problem appears under different names in the literature. Dyckhoff [4] lists some examples:

- cutting stock and trim loss problems;
- bin packing, dual bin packing, strip packing, vector packing, and knapsack (packing) problems;
- vehicle loading, pallet loading, container loading, and car loading problems;
- assortment, depletion, design, dividing, layout, nesting, and partitioning problems;
- capital budgeting, change making, line balancing, memory allocation, and multi-processor scheduling problems.

While the problems listed above have different names and applications, they share a logical structure. In particular, all have two fundamental properties (Dyckhoff, 1990):

1. There are two groups of basic data whose elements define geometric bodies of fixed shapes in a one- or more-dimensional space of real numbers. The group of large objects is called ‘the stock’; the other group of small objects (called items) is the list of ordering or required items.
2. The cutting or packing process performs geometric allocation of small items to large objects. Residual space not enough for small items is usually called ‘trim loss’.

Since these two characteristics of the cutting and packing problem are common in many situations, the problem can be extended to cover many problems sharing the same properties in the abstract dimensions.

1. Dimension of weight: knapsacking, and vehicle loading.
2. Dimension of time: assembly line balancing and multiprocessor scheduling.
3. Financial dimension: capital budgeting and change making.
4. Other dimension: computer memory allocation.

CSPs can be efficiently classified by using the classification scheme developed by Dyckhoff. Since there is a strong relationship between cutting and packing, the scheme is applied to both
problem categories. This relationship results from the duality of material and space, the
duality of solid material body and the space occupied by it. In a certain sense, cutting can be
seen as packing the space occupied by small items into the space occupied by large objects.
On the other hand, packing can be seen as cutting the empty space of the large objects into
parts of empty spaces some of which are occupied by small items, the other being trim loss.
The classification of the problems can be divided into four characteristics.

1. Dimensionality
   (1) One-dimensional.
   (2) Two-dimensional
   (3) Three-dimensional
   (N) N-dimensional with N>3

2. Kind of assignment
   (B) All objects (stocks) and a selection of items
   (V) A selection of objects and all items.

3. Assortment of large objects
   (O) One object.
   (I) Many objects of identical figure
   (D) Different figures

4. Assortment of small items
   (F) Few items (of different figures).
   (M) Many items of many different figures.
   (R) Many items of relatively few different (non-congruent) figures.
   (C) Many identical items.

By combining the distinguished major types of the above four characteristics, many cutting
and packing problems can be classified into 96 different categories. For example, 3/B/O/F
denotes all three-dimensional cutting and packing problems where one large object has to be
packed with a selection out of a few small items. Using Dyckhoff's classification scheme, the
classic knapsack problem is one-dimensional with one large object that has to be packed with
a selection from the set of small items (1/B/O/). The pallet loading problem is two-
dimensional with identical small items (2/B/O/C). Other examples are: the classic bin packing
problem (1/V/I/M), the classic cutting stock problem (1/V/I/R), and the assembly line
balancing problem (1/V/I/M).
The most important characteristic is dimensionality. The dimensionality is the minimum number of dimensions that are significant in the determination of the solution. Dyckhoff lists the elementary types of dimensionality as one-, two-, three- and multidimensional problems. However, a number of CSPs with complexity between one and two-dimensional problems exist and these can be characterized as 1.5-dimensional problems (Hinxman, [11]; Haessler & Sweeney [10]).

In the simplest case, only one dimension of the stock is relevant to the solution. A typical example of the one-dimensional CSP is the cutting of steel bars where the length of the stock bars is fixed. In 1.5-dimensional problems the stock has one fixed and one variable dimension. In two-dimensional problems the ordered items can be positioned on the stock material in two free dimensions. The simplest case arises when both the stock sheet and the ordered items are rectangular.

In two-dimensional problems, the cutting process may consist of distinct steps in which the stock material is cut to sub-sheets at right angles to the cuts in the previous step. This kind of cutting process will be called staged two-dimensional cutting. If the cuts should be made parallel to a side of the stock sheet, the problem will be called as orthogonal 2-dimensional. Finally, if the cut must extend the full width of the sheet or some sub-sheet produced by previous cut, it will be called guillotine cut (Hinxman, [11]).

Different cutting procedures are illustrated in Figure 4.1. Hereafter, grey areas represent trim loss. In a one-dimensional CSP illustrated in Figure 4.1a width of the stock reel B is fixed while in the 1.5-dimensional case in Figure 4.1b the width is variable. Notice that in both cuttings, pattern lengths are fixed. In two-dimensional CSP illustrated in Figure 4.1c cutting pattern lengths are variable. In Figures 4.1a–c cuttings are orthogonal two-staged guillotine, while in Figure 4.1d cuttings are non-guillotine.
Another important characteristic affecting to the complexity of the problem is the kind of assignment. For the classification of the particular CSP, at least two categories are needed. In the first one, all stock material will be used, but not all the orders must be fulfilled. In the second one, every order must be fulfilled, but only a portion of stock will be used.

The third characteristic can be divided into three types with respect to the assortment of large objects. First, there is only one large object to be cut. In the second type, there are many large objects with similar dimensions, and finally, there are many large objects with different dimensions. For example, the last case occurs when the stock contains residual pieces of previous cutting periods.

Similar division of types can be made also to the assortment of small items. In the first type there are only relatively few small items with different dimensions. In the second type there are many small items, most of them having different dimensions. In the third type there are many small items, most of them having the same dimensions. In the fourth type all the small items have identical dimensions.
4.4 Approach of the waste reduction problem

As seen in Chapter 3, everything begins with drawing the veranda, making a production order and recalculating the presupposed price, all this in the engineering firm. The sawing step is the first production step, in the roof department, after taking the required profiles out of the warehouse.

Nowadays, the company just takes up the required profiles for one order, i.e. one veranda project, out of the warehouse. As all the profiles are seven meters long, you can imagine that this way of production causes a lot of waste. It often happens that they need to cut for example only one piece of three meters or two pieces of two meters, which results in a huge waste.

Significant is the fact that we are not speaking of brute material. The waste material is aluminium roof profiles that have been extruded, treated. In most cases, the profiles are provided of a thermal break: the ridge-tile, the gutter and the intermediate rafter for the Confort and Prestige+ and the lateral rafter only for the Prestige+ profiles (see Chapter 1 and sketches in Appendix B).

Aluminium roof profiles are expensive; so, the maximum waste reduction is important. For example, a lacquered aluminium profile costs approximately 5 €/kg; waste is sold at 1 €/kg, which gives a difference of 4 €/kilo. For aluminium profiles, waste is in the range of 25 %. “If we can decrease the waste until 23 %, we immediately feel the difference”, says the company.

To restrict the waste the remained material can be restocked, what they are actually doing, but because of the vast numbers of profiles multiplied by the immense variety of colours offered this can not be done for all combinations. The profiles which are not restocked are obviously not thrown away. They are stored in two different containers: one for profiles with thermal break and another for profiles without. The more expensive profiles, with thermal break, have a smaller total aluminium percentage due to the thermal break itself which is made of polyamide reinforced with glass fibre, and are sold for a lower price than the profiles without thermal break.
The idea is to regroup multiple veranda projects, with the objective to minimize the total waste. We will do this regrouping in different steps. In a first step the BOM (Bill Of Material) with regard to the roof profiles of all veranda projects over a certain period will be made. The company is currently working with FoxPro as programming language. It is possible to export a list from this system with the different orders and extract the required BOM.

Subsequently, we will divide this long list, of profiles to be cut, in different groups according to their colour. In the following step, a further subdivision is made for the different profiles. Now we have packages of same profile and colour, which are ready to be optimized. All the profiles on the same list have to be cut from the same stock profile. This cutting stock problem is an optimization problem, and more specifically an integer linear programming problem.

The above method will now be illustrated with real data. To keep it conveniently arranged, we will work with a little example of just two veranda projects with filename: 051481 and 051455. Of course it is possible to repeat the same procedure for a lot of veranda projects. The complete file of the project 051481 can be consulted in Section 2.3. In the first step the BOM is made, which gives an overview of all necessary profiles, their lengths, and all requirements. For the cutting stock problem we will only work with the roof profiles, as these are the profiles that are being cut in the sawing area. In the following step we make a list, dividing the roof profile BOM in different groups according to their colour. In this illustrative example we took two projects of the same colour (RAL 9010), so this step is skipped. Subsequently the list is regrouped on roof profile type. Now for every profile type we can run the optimization program. The following list (table 4.1) gives the result of these steps for the projects 051481 and 051455, where length is measured in mm.

<table>
<thead>
<tr>
<th>Number of pieces</th>
<th>Colour</th>
<th>Length</th>
<th>Number</th>
<th>File</th>
<th>Max. Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>E 02/25</td>
<td>9010</td>
<td>1948</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
<tr>
<td>E 02/25</td>
<td>9010</td>
<td>2941</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>I 43x27x2</td>
<td>9010</td>
<td>1912</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>L 100x25x2</td>
<td>9010</td>
<td>5000</td>
<td>1</td>
<td>051455</td>
<td>5950</td>
</tr>
<tr>
<td>L 50x25x2</td>
<td>9010</td>
<td>5000</td>
<td>1</td>
<td>051455</td>
<td>5950</td>
</tr>
<tr>
<td>Material</td>
<td>Code</td>
<td>Quantity</td>
<td>Unit</td>
<td>Part Number</td>
<td>Finish</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>----------</td>
<td>------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>P 30x2</td>
<td>9010</td>
<td>2000</td>
<td>1</td>
<td>051481</td>
<td>5950</td>
</tr>
<tr>
<td>P 30x2</td>
<td>9010</td>
<td>2000</td>
<td>1</td>
<td>051481</td>
<td>5950</td>
</tr>
<tr>
<td>P 30x2</td>
<td>9010</td>
<td>2000</td>
<td>1</td>
<td>051481</td>
<td>5950</td>
</tr>
<tr>
<td>P 30x2</td>
<td>9010</td>
<td>2500</td>
<td>1</td>
<td>051455</td>
<td>5950</td>
</tr>
<tr>
<td>P 30x2</td>
<td>9010</td>
<td>2500</td>
<td>1</td>
<td>051455</td>
<td>5950</td>
</tr>
<tr>
<td>P 30x2</td>
<td>9010</td>
<td>2500</td>
<td>1</td>
<td>051455</td>
<td>5950</td>
</tr>
</tbody>
</table>

**SOLIN 01**

<table>
<thead>
<tr>
<th>Material</th>
<th>Code</th>
<th>Quantity</th>
<th>Unit</th>
<th>Part Number</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLIN 01</td>
<td>9010</td>
<td>3250</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
</tbody>
</table>

**TRAVERSE 01**

<table>
<thead>
<tr>
<th>Material</th>
<th>Code</th>
<th>Quantity</th>
<th>Unit</th>
<th>Part Number</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAVERSE 01</td>
<td>9010</td>
<td>954</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>TRAVERSE 01</td>
<td>9010</td>
<td>954</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>TRAVERSE 01</td>
<td>9010</td>
<td>954</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>TRAVERSE 01</td>
<td>9010</td>
<td>954</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>TRAVERSE 01</td>
<td>9010</td>
<td>1030</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
</tbody>
</table>

**VER 01/25A**

<table>
<thead>
<tr>
<th>Material</th>
<th>Code</th>
<th>Quantity</th>
<th>Unit</th>
<th>Part Number</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>VER 01/25A</td>
<td>9010</td>
<td>1924</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
<tr>
<td>VER 01/25A</td>
<td>9010</td>
<td>1924</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
<tr>
<td>VER 01/25A</td>
<td>9010</td>
<td>2918</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>VER 01/25A</td>
<td>9010</td>
<td>2918</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>VER 01/25A</td>
<td>9010</td>
<td>2918</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
</tbody>
</table>

**VER 02/25**

<table>
<thead>
<tr>
<th>Material</th>
<th>Code</th>
<th>Quantity</th>
<th>Unit</th>
<th>Part Number</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>VER 02/25</td>
<td>9010</td>
<td>1924</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
<tr>
<td>VER 02/25</td>
<td>9010</td>
<td>1924</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
<tr>
<td>VER 02/25</td>
<td>9010</td>
<td>2918</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>VER 02/25</td>
<td>9010</td>
<td>2918</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
</tbody>
</table>

**VER 05/25C**

<table>
<thead>
<tr>
<th>Material</th>
<th>Code</th>
<th>Quantity</th>
<th>Unit</th>
<th>Part Number</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>VER 05/25C</td>
<td>9010</td>
<td>3250</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
<tr>
<td>VER 05/25C</td>
<td>9010</td>
<td>5000</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
</tbody>
</table>

**VER 06**

<table>
<thead>
<tr>
<th>Material</th>
<th>Code</th>
<th>Quantity</th>
<th>Unit</th>
<th>Part Number</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>VER 06</td>
<td>9010</td>
<td>934</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>VER 06</td>
<td>9010</td>
<td>934</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>VER 06</td>
<td>9010</td>
<td>946</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>VER 06</td>
<td>9010</td>
<td>946</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>VER 06</td>
<td>9010</td>
<td>1010</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
<tr>
<td>VER 06</td>
<td>9010</td>
<td>1022</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
</tbody>
</table>

**VER 11/10**

<table>
<thead>
<tr>
<th>Material</th>
<th>Code</th>
<th>Quantity</th>
<th>Unit</th>
<th>Part Number</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>VER 11/10</td>
<td>9010</td>
<td>275</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
<tr>
<td>VER 11/10</td>
<td>9010</td>
<td>275</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
<tr>
<td>VER 11/10</td>
<td>9010</td>
<td>287</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>VER 11/10</td>
<td>9010</td>
<td>287</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>VER 11/10</td>
<td>9010</td>
<td>1868</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
<tr>
<td>VER 11/10</td>
<td>9010</td>
<td>1868</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
<tr>
<td>VER 11/10</td>
<td>9010</td>
<td>1878</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
<tr>
<td>VER 11/10</td>
<td>9010</td>
<td>1878</td>
<td>1</td>
<td>051481</td>
<td>6950</td>
</tr>
<tr>
<td>VER 11/10</td>
<td>9010</td>
<td>2858</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
<tr>
<td>VER 11/10</td>
<td>9010</td>
<td>2858</td>
<td>1</td>
<td>051455</td>
<td>6950</td>
</tr>
</tbody>
</table>
4.5 The Cutting Stock Problem

4.5.1 An example

Let us first work out an example to make things clear. Suppose we have to cut all required parts out of a minimum number of stock profiles with length 100. Demand is as follows:

<table>
<thead>
<tr>
<th>Order Length</th>
<th>Quantity Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4-2 Example 1 CSP
This is of course a very easy example, where we immediately can see that it is possible to cut all the required parts out of two stock profiles, as follows: 50 – 25 – 25 and 30 – 20 – 10 - 10 - 10. When the ordered quantities are higher, we will not be able to see it immediately, and need an algorithm. When one has to solve this with an algorithm, the first thought would be to use a ‘Greedy Algorithm’.

*Begin with the biggest part, cut as much as you can until the ordered quantity is achieved or until you can not put any more of them in the remained stock material. Next you go to the second biggest order length and do the same thing, cutting as much parts as possible out of the remained stock material until one of the two stop conditions occurs. You do this until the smallest order length. If all demands are fulfilled, you stop; if not, take another stock material and repeat the same loop.*

This greedy algorithm is very easy to implement and will give a good solution for a lot of easy cases, but it is not a good algorithm. The following easy case illustrates this. Suppose again that we have to cut all required parts out of a minimum number of stock profiles with length 100. Demand is as follows:

<table>
<thead>
<tr>
<th>Order Length</th>
<th>Quantity Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>1</td>
</tr>
<tr>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
</tr>
</tbody>
</table>

*Table 4-3  Example 2 CSP*

With the greedy algorithm we would find:  
Stock material 1: 70 – 20
Stock material 2: 60 – 20 – 15
Stock material 3: 15

With this algorithm we need three stock profiles. As this is an easy example, we are able to see that it was possible to cut it out of two stock profiles, as follows: 60 – 20 – 20 and 70 – 15 – 15.

As it was already said, this cutting stock problem is an optimization problem, and more specifically an integer linear programming problem, so it is possible to write it down as a mathematical model.
4.5.2 Mathematical model

Optimization problems can be solved through Linear Programming (LP), in which the objective function and the constraints are all linear. If all unknown variables have to be integers, then it becomes Integer Programming (IP) or Integer Linear Programming (ILP). In contrast to linear programming, which can be solved efficiently in the worst case, integer programming problems are in the worst case undeterminable, and in situations with bounded variables NP-hard (Non-deterministic Polynomial-time hard).

Let the variable $x_j$ be the number of each pattern $j$ to cut. A pattern is a possible way of cutting the stock material, like 70 – 20 or 60 – 20 – 20 or 60 – 15 – 15 or 20 – 15 – 15. The objective function is to minimize the number of stock profiles to be cut. The constraints in the problem require that enough stock material is cut with certain patterns to fulfil the received orders. In our example, the objective and the constraints can easily be formulated as mathematical equations.

Let $m$ be the number of order lengths in the model. Let $n$ be the number of patterns in the model. Let $a_{ij}$ be the number of pieces cut of the $i^{th}$ order length out of the $j^{th}$ pattern. Let $b_i$ be the demand for the $i^{th}$ length.

If $x_j$ is the number of times the $j^{th}$ pattern will be used, then for every order length $i$, $a_{ij} \times x_j$ must be greater than or equal to the required number of pieces for this order length. This is the main set of constraints. What we want to minimize is the number of stock profiles, which is the sum of all the variables $x_j$; this is the objective function. Another set of constraints is that all the variables $x_j$ have to be non-negative and integer. There can never be a negative number of patterns neither a mix of different patterns used, where half of one pattern and half of other pattern is used. This last set of constraints is obvious, but important.

$$
\text{Min} \quad \sum_{j}^{n} x_j \\
\text{Subject to} \quad \sum_{j}^{n} a_{ij} x_j \geq d_i \quad \forall i \\
x_j \geq 0 \text{ and int} \quad \forall j
$$
Integer linear programs are difficult to solve, and a usual technique for solving integer linear programming problems is to just forget about the integer part; this is called relaxation. Later, we may use some integer programming techniques to ensure we get an integer solution. We now have to solve a linear programming problem. This is a common optimization problem for which good techniques exist for the resolution. One method is the simplex method (duality), developed by George Dantzig, which starts from a non-optimal point and iteratively finds better and better solutions. It constructs an admissible solution at a vertex of the polyhedron (feasible region defined by the linear constraints), and then moving along edges of the polyhedron to vertices with successively better values of the objective function until the optimum is reached. The algorithm concludes with the optimal solution when no additional iteration can be made to improve the solution. The simplex method guarantees to find the global optimum if certain precautions against cycling are taken.

To formulate the linear program, we have to create all possible patterns that might be cut from a stock profile. If there are many different order lengths to cut from the stock profiles, there may be an exponential number of patterns that need to be included in the linear program.

To illustrate this, we can create all possible cutting patterns for the example 2 above, stock profiles of length 100 and following demands: one of 70, one of 60, two of 20 and two of 15. The different possible cutting patterns are:

70
70 20
70 15
70 15 15
60
60 20
60 20 20
60 20 15
60 15
60 15 15
20
20 20
20 20 20
20 20 20 20
20 20 20 20 20
20 20 20 20 15
20 20 20 15
20 20 20 15 15
20 20 15
For this little example with only four different order lengths we already have 33 possible patterns. In this example with very low demands we can drop some patterns because more pieces are cut than needed. If we do this, 18 cutting patterns remain.

Because the number of feasible cutting patterns, \( n \), can be enormous, the problem is difficult in computational terms. However, the problem can be solved efficiently without first enumerating every feasible cutting pattern. One alternative is to use a Delayed Column Generation approach, as shown in Gilmore & Gomory (1961, 1963).

### 4.5.3 Delayed column generation

Delayed column generation method solves the cutting stock problem by starting with just a few patterns. It generates additional patterns when they are needed. We choose an initial set of patterns to include in the model and solve the linear program. Since it is unlikely that we chose the right set of patterns, we use the dual variable information from the linear program to generate a new pattern. (These dual variables take the same value as the shadow prices in the primal. The shadow price is the improvement in the objective value of the optimal solution obtained by increasing the right hand side of the constraint by one unit. When solving the LP with the simplex method, the shadow prices are generated.)

These new patterns will reduce the number of stock profiles used. New patterns are generated using the knapsack problem. It derives its name from the maximization problem of choosing as much as possible essentials that can fit into one bag (of maximum weight) to be carried on
a trip. This is another optimization problem, which can be rewritten as follows, if a general pattern \( j \) is considered:

\[
\begin{align*}
\text{Min} & \quad 1 - \sum_{i}^{m} \pi_{i} a_{ij} \\
\text{Subject to} & \quad \sum_{i}^{m} a_{ij} l_{i} \leq L \\
& \quad a_{ij} \geq 0 \quad \text{and} \quad \text{int} \quad \forall i
\end{align*}
\]

With \( \pi_{i} \) = the shadow price of constraint \( i \) of the LP for the CSP

\( L = \) length of stock profiles

\( m = \) number of order lengths

The knapsack problem considers all potential cutting patterns, since a cutting pattern is feasible whenever it satisfies the knapsack constraint, and determines the coefficients \( a_{ij} \) of a new cutting pattern. The optimal values for \( a_{ij} \) indicate how many of each length \( l_{i} \) should be included in the new cutting pattern \( j \).

The knapsack problem has well-known methods to solve it, among which are branch and bound and dynamic programming. The delayed column generation method turns out to be much more efficient than the original approach.

These two problems, the main linear program and the knapsack problem, are solved in turn until no more patterns, which will reduce the number of rolls cut, can be found. This will be the case if the reduced costs of the master problem are positive! The reduced cost is:

\[
1 - \sum_{i} \pi_{i} a_{ij}
\]

So if the optimal solution of the knapsack problem, which is the minimum of the reduced costs, results in a positive number, we can conclude that with the generated patterns an optimal solution for the CSP has been found. In a last step the, master problem is solved again without the relaxation, so as an integer linear problem (ILP).
4.5.4 Procedure

Let us briefly review the whole procedure.

1. Find an initial set of patterns for the master problem by generating $n$ patterns each containing the largest possible integer number of one ordered size.

   $a_{ij} = \left\lfloor \frac{L}{l_i} \right\rfloor$ for $i = 1 \ldots m$

   $a_{ij} = 0$ for $i \neq j$, $i = 1 \ldots m$, $j = 1 \ldots m$

2. Solve the master problem with the simplex algorithm.

   $\text{Min} \quad \sum_{j}^{n} x_j$

   $\text{Subject to} \quad \sum_{j}^{n} a_{ij} x_j \geq d_i \quad \forall i \quad \pi_i$

   $x_j \geq 0 \quad \forall j$

3. Solve the knapsack problem by using the shadow prices ($\pi_i$) of the linear problem.

   $\text{Min} \quad 1 - \sum_{i}^{m} \pi_i a_{ij}$

   $\text{Subject to} \quad \sum_{i}^{m} a_{ij} l_i \leq L$

   $a_{ij} \geq 0 \quad \text{and} \quad \text{int} \quad \forall i$

4. If the objective value of the knapsack problem is negative, go to step 2 and add the new pattern to the linear problem.

5. Solve the master problem with all generated patterns and without relaxation; an integer linear problem.
4.6 Improvement of column generation algorithm

Above algorithm works very good if the demands are high. As it is a heuristic, we will in most cases not get the optimal solution but a solution near the optimal one. If the demands are high this near optimal solution will be good enough, because the fault in terms of percentage will be small enough to accept it.

4.6.1 The Cutting Stock Problem with low demands

For the specific case of the company, demands are the number of profiles ordered with the same length. As the lengths are measured up to the millimetre and all verandas are different, it is clear that the quantities of the demands will be low. Rounding optimal linear programming solutions to integers does not lead to good results. In some cases, the optimal integer solution of the CSP can not be found starting from the optimal non-relaxed solution.

To illustrate this, we will solve example 1 with Gilmore and Gomory’s algorithm. We already noticed that the solution of the Greedy Algorithm for this example was wrong.

<table>
<thead>
<tr>
<th>Order Length</th>
<th>Quantity Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4-4 Repetition of example 1 CSP

In step 1 of the algorithm the initial patterns are made:

50 – 50
30 – 30 – 30
20 – 20 – 20 – 20 – 20

These five patterns are presented by the following matrix A in the algorithm:
After solving the master problem for the first time, we get the initial shadow prices which are 0.5, 0.333, 0.25, 0.2 and 0.1 for respectively the constraints with order length 50, 30, 25, 20 and 10. The objective value is 2.033. Solving the knapsack problem gives a new pattern: 30 – 30 – 30 – 10, which is a good pattern since the waste is zero. This pattern is presented by a new column in our matrix A:

\[
\begin{bmatrix}
2 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 & 0 \\
0 & 0 & 4 & 0 & 0 \\
0 & 0 & 0 & 5 & 0 \\
0 & 0 & 0 & 0 & 10
\end{bmatrix}
\]

The objective function of the knapsack problem is -0.099, which is negative; so, we go back to step 2. The objective function is already close to zero, so few more loops are expected.

Solving the master problem for the second time gives following shadow prices: 0.5, 0.3, 0.25, 0.2 and 0.1, in the same order as above. The objective value is 2. Solving the knapsack problem gives an objective value of zero, which means no new patterns will improve the objective function of the relaxed master problem.

In step 5 of the algorithm, the master problem is solved again with the six patterns found but without the relaxation, i.e. with an integer solution. The relaxed solution is:

\[
\begin{bmatrix}
0.5 & 0 & 0.5 & 0.4 & 0.267 & 0.333
\end{bmatrix}
\]

, which means 0.5 times pattern 1 represented by column one in the matrix; 0 times pattern 2; 0.5 times pattern 3 and so on... This solution gives an objective function of 2, which is the optimal non-relaxed solution. We already see that rounding this solution will give some problems. With the branch-and-bound algorithm, the following integer solution is found:

\[
\begin{bmatrix}
1 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

, which gives an objective function of 5.

When all \(x_j\) are smaller than one we will almost always get a complete wrong result. As seen in Section 4.5.1, the optimal integer solution is 50 – 25 – 25 and 30 – 20 – 20 – 10 - 10 - 10.
With this solution we use only two stock profiles instead of five, the answer of the Gilmore and Gomory’s algorithm.

### 4.6.2 Low Demand Column Generation Algorithm

Above section showed that Gilmore and Gomory’s algorithm cannot be used for our problem. As the size of the cutting stock problem is small (CSP with more than 1000 variables are solved in the literature), an eventually slower but more efficient algorithm is needed.

Scheithauer et al. [13] present an exact solution approach for the one-dimensional cutting stock problem. The solution is based on a combination of the cutting plane method and the column generation technique. Results of extensive computational experiments are reported. Another exact solution is presented by De Carvalho [3].

These two exact procedures are not made especially for the kind of problem we are studying, so we opt for adapting Gilmore and Gomory’s algorithm for the particular problem in this company. In these exact solutions is found by changing the manner of finding new branch and bound nodes in the knapsack problem.

In first instance we will adapt the knapsack problem. In the original algorithm the knapsack problem has only one constraint which says that the summation of the length of the pieces used in a pattern must be smaller than the total stock profile length. The knapsack problem does not look at the demand to make new patterns. In the above example (CSP 1) we need only one piece of length 30, but the first pattern found solving the knapsack problem is 30 – 30 – 30 – 10. At first sight this pattern seems to be a good since the waste is zero. But in this pattern three pieces of length 30 will be cut and if we look at the demand we only need one piece of length 30. This is the reason why in some cases in the end we get an optimal relaxed solution where some patterns are used less than one time, as seen in CSP 1.

Extending the knapsack problem with some extra constraints can prevent this. We will make sure that in every new pattern the number of cut pieces of a particular length are smaller than the total number of pieces needed. In the algorithm the following constraints will be added:

\[ a_{ij} \leq d_i \quad \forall \ i \]
In a CSP with five different lengths, the knapsack problem will be extended to six constraints. It is clear that the new knapsack problem will be much more difficult to solve, especially as it is an integer optimization problem. We will have to simplify the new knapsack problem. This can be done by working with an upper bound on each variable, so all the new added constraints can be deleted.

Now we will have new patterns in which the number of cut pieces of a particular length is smaller than the total number of pieces needed of that particular length. But the initial patterns do not fill this, so we will have to change them too. Let \( a_{ii} \) be the minimum of the integer part of the total profile length divided by the \( i^{th} \) order length and the \( i^{th} \) order demand, for any order length. The diagonal elements of the matrix with the adapted initial patterns will be:

\[
a_{ii} = \min \left\{ \left\lfloor \frac{L}{L_i} \right\rfloor \text{ and } d_i \right\} \quad \text{for } i = 1 \ldots m
\]

The complete low demand column generation algorithm is as follows:

1. Find an initial set of patterns for the master problem by generating \( m \) patterns.

\[
a_{ii} = \min \left\{ \left\lfloor \frac{L}{L_i} \right\rfloor \text{ and } d_i \right\} \quad \text{for } i = 1 \ldots m
a_{ij} = 0 \quad \text{for } i \neq j, i = 1 \ldots m, j = 1 \ldots m
\]

2. Solve the master problem with the simplex algorithm.

\[
\begin{align*}
\text{Min} & \quad \sum_j^n x_j \\
\text{Subject to} & \quad \sum_j^n a_{ij} x_j \geq d_i \quad \forall i \quad \pi_i \\
& \quad x_j \geq 0 \quad \forall j
\end{align*}
\]

3. Solve the knapsack problem by using the shadow prices \((\pi_i)\) of the master problem.

\[
\begin{align*}
\text{Min} & \quad 1 - \sum_i^m \pi_i a_{ij} \\
\text{Subject to} & \quad \sum_i^m a_{ij} l_j \leq L \\
& \quad 0 \leq a_{ij} \leq d_i \quad \text{and} \int \quad \forall i
\end{align*}
\]
4. If the objective value of the knapsack problem is negative, go to step 2 and add the new pattern to the master problem.

5. Solve the master problem with all generated patterns and without relaxation, i.e. an integer linear problem.

### 4.7 AMPL program

AMPL is a comprehensive and powerful algebraic modelling language for linear and nonlinear optimization problems, in discrete or continuous variables. It was developed at Bell Laboratories. On the website http://www.ampl.com/ a free student edition with some restrictions can be downloaded.

In this section we will run the low demand column generation algorithm with AMPL using an example with real data. Because AMPL is built especially to program optimization problems it is an easy programming language and convenient to demonstrate the algorithm. Unfortunately the program is not compatible with the company’s system. In Chapter 5, an easy-to-use program will be developed.

The AMPL program consists of three files: a .run file which contains the program, a *.mod file which contains the model and a *.dat file which contains the data.

#### 4.7.1 csp.run file

```AMPL
option solver cplex;
option solution_round 6;

model csp.mod;
data csp.dat;

problem Cutting_Stock_Problem: Cut, Number, Fill;
    option relax_integrality 1;
    option presolve 0;

problem Pattern_Generation: Use, Reduced_Cost, Length_Limit, Demand_Limit;
    option relax_integrality 0;
    option presolve 1;

let nPAT := 0;
```
for \( i \) in LENGTHS \{ 
    let nPAT := nPAT + 1;
    let nbr\[i,nPAT\] := min \{ \text{floor} (\text{profile length}/i), orders[i]\};
    let \{i2 in LENGTHS: i2 <> i\} nbr\[i2,nPAT\] := 0;
\};

for \( i \) in LENGTHS \{ 
    let nPAT := nPAT + 1;
    let nbr\[i,nPAT\] := \text{floor} (\text{profile length}/i);
    let \{i2 in LENGTHS: i2 <> i\} nbr\[i2,nPAT\] := 0;
\};

repeat {
    solve Cutting_Stock_Problem;
    let \{i in LENGTHS\} price[i] := Fill[i].dual;

    solve Pattern_Generation;

    if Reduced_Cost < -0.00001 then {
        let nPAT := nPAT + 1;
        let \{i in LENGTHS\} nbr\[i,nPAT\] := Use[i];
    } else break;
};

display nbr;
display Cut;

option Cutting_Stock_Problem.relax_integrality 0;
option Cutting_Stock_Problem.presolve 10;
solve Cutting_Stock_Problem;
display Cut;

### 4.7.2 csp.mod file

```plaintext
# --------------
# MASTERPROBLEM
# --------------

param profile_length > 0;

set LENGTHS;
param orders {LENGTHS} > 0;

param nPAT integer >= 0;
set PATTERNS := 1..nPAT;

param nbr {LENGTHS,PATTERNS} integer >= 0;

    check {j in PATTERNS}:
        sum {i in LENGTHS} i * nbr[i,j] <= profile_length;

var Cut {PATTERNS} integer >= 0;

minimize Number:
```

83
Chapter 4  Material Waste Problem

sum {j in PATTERNS} Cut[j];

subj to Fill {i in LENGTHS}:
    sum {j in PATTERNS} nbr[i,j] * Cut[j] >= orders[i];

# ----------------------------------------
# KNAPSACK SUBPROBLEM FOR CUTTING STOCK
# ----------------------------------------

param price {LENGTHS} default 0.0;

var Use {LENGTHS} integer >= 0;

minimize Reduced_Cost:
    1 - sum {i in LENGTHS} price[i] * Use[i];

subj to Length_Limit:
    sum {i in LENGTHS} i * Use[i] <= profile_length;

subj to Demand_Limit {i in LENGTHS}:
    Use[i] <= orders[i];

4.7.3  csp.dat file

As an example we will run the program with some real data. The following data can be found in Section 4.4 for the profile VER11/10:

<table>
<thead>
<tr>
<th>Order Length</th>
<th>Quantity Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1868</td>
<td>2</td>
</tr>
<tr>
<td>275</td>
<td>2</td>
</tr>
<tr>
<td>1878</td>
<td>2</td>
</tr>
<tr>
<td>287</td>
<td>2</td>
</tr>
<tr>
<td>2874</td>
<td>2</td>
</tr>
<tr>
<td>2858</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4-5  Example 3 CSP

Translating this in AMPL-language gives following csp.dat file:

param profile_length := 6950 ;

param: LENGTHS: orders :=
    1868 2
    275 2
    1878 2
    287 2
    2874 2
    2858 2;
4.7.4 Result

In contrast with previous example, this example with real data is at first sight not so easy to solve. The AMPL program gives following result:

```
CPLEX 8.0.0: optimal solution; objective 3.496666667
0 dual simplex iterations (0 in phase I)
CPLEX 8.0.0: optimal integer solution; objective -0.208333
25 MIP simplex iterations
24 branch-and-bound nodes
CPLEX 8.0.0: optimal solution; objective 3.288333333
1 simplex iterations (0 in phase I)
CPLEX 8.0.0: optimal integer solution; objective -0.208333
7 MIP simplex iterations
10 branch-and-bound nodes
CPLEX 8.0.0: optimal solution; objective 3.08
1 simplex iterations (0 in phase I)
CPLEX 8.0.0: optimal integer solution; objective -0.163334
1 MIP simplex iterations
0 branch-and-bound nodes
CPLEX 8.0.0: optimal solution; objective 3.08
1 simplex iterations (0 in phase I)
CPLEX 8.0.0: optimal integer solution; objective -0.163334
2 MIP simplex iterations
0 branch-and-bound nodes
CPLEX 8.0.0: optimal solution; objective 3
2 simplex iterations (0 in phase I)
CPLEX 8.0.0: optimal integer solution; objective -0.04
13 MIP simplex iterations
8 branch-and-bound nodes
CPLEX 8.0.0: optimal solution; objective 3
1 simplex iterations (0 in phase I)
CPLEX 8.0.0: optimal integer solution; objective 0
1 MIP simplex iterations
0 branch-and-bound nodes
nbr [*,*] (tr)
:     275   287  1868  1878  2858  2874    :=
  1       0     0     2     0     0     0
  2       2     0     0     0     0     0
  3       0     0     0     2     0     0
  4       0     2     0     0     0     0
  5       0     0     0     0     2     0
  6       0     0     0     0     2     0
  7       0     0     3     0     0     0
  8     25     0     0     0     0     0
  9       0     0     0     3     0     0
 10      0     24    0     0     0     0
 11      0     0     0     0     0     2
 12      0     0     0     2     0     0
 13      0     1     2     0     1     0
 14      0     1     0     2     1     0
 15      2     2     0     0     2     0
 16      2     2     0     0     0     2
 17      1     0     2     0     1     0
;
Cut [*] :=
```
We observe in the AMPL output that the optimal relaxed solution is an integer value, so it is the same as the non relaxed solution. For most of the problems this will not be the case. 17 patterns are generated. In the AMPL output the patterns are showed in rows instead of columns.

We need 3 stock profiles with the following patterns:

2858 – 1868 – 1868 – 287
2858 – 1878 – 1878 – 287
The order length 287 is cut four times, but it is only needed two times. This is a consequence of the algorithm. The constraints in the master problem say that the demand must be fulfilled using the sign ‘greater or equal to’. In some cases, like this one, the demand will be more than fulfilled as the combination of the best patterns found by solving the knapsack problem results in a surplus of a certain order length.

4.8 Discussion of specific CSP for the veranda industry

In this section we will further discuss the specific cutting stock problem of the company and study possible additional improvements. The following pictures of the sawing machine give a good view.

![Figure 4-2 The sawing machine used in the company](image-url)
This is the sawing machine (Megal, SW 400 M), which is currently used by the company to cut the aluminium profiles. The machine can bend horizontally and vertically. A first limitation of the problem is the space the company has at its disposal. All profiles of one veranda project are put on a cart. We saw that only three carts with veranda profiles can be placed next to each other, between the multiple sawing machines. The profiles are placed horizontally on the cart, whereby they use a lot of place. Because of stability reasons and safety, they can not be stocked vertically.

In the future an automatic sawing machine can be purchased, which will increase the number of veranda orders that can be combined. The patterns generated by the algorithm with their multiplication factor will be electronically inputted to the machine. Each profile will get a label with the veranda file number.

As can be seen in the profiles BOM in Section 2.3, all the pieces to be cut have a given length and some of them also a vertical or horizontal angle. When these pieces are cut a triangle remains, so we get a little bit more wasting than with straight cuts. We inquired the company about the possibility of combining two pieces with an angle, so that the waste would be recuperated. Unfortunately, this combination is not possible in practice, as the profiles are not symmetric and can not be turned on the table.

Another possible improvement can be made if we would place two profiles on each other. If a pattern has to be cut more than once, placing two profiles on each other would accelerate the
procedure. This is also not possible for the same reason. One bottom of the profile is mostly flat, but not the other side, so that two profiles can not be put on each other.

A last research was made to see if it is possible to put two different pieces on each other. This is possible for some profiles that can be telescoped into each other. The pieces are always of the same length, so the patterns will also be the same, which make a combination easy and gaining time.

4.9 Exact Algorithm

The low demand column generation algorithm is an improvement of Gilmore and Gomory’s algorithm for low demands and gives good results. But because of the delayed column generation it is still a heuristic. There will be problems where an optimal solution of the non-relaxed master problem will be obtained, without finding all the patterns that will give the best solution of the ‘integer-solved’ master problem.

In this section an algorithm that gives an exact solution will be developed without changing the Branch-and-Bound code, as required in previous mentioned exact algorithms.

For the particular case of the company, demands will always be low, as all verandas are different. Keeping this in mind, an algorithm that will first generate the patterns can be developed. The patterns will be generated on a smart way, so that not ‘all’ patterns are generated but only a little portion of it. Instead of generating the patterns by solving the knapsack problem (delayed column generation), all ‘full’ and ‘required’ patterns will be generated in the beginning of the algorithm.

A ‘full’ pattern is a pattern in which no more pieces can be added. In Section 4.5.2 all patterns for example 2 were generated. The pattern 60 and 60 – 20 are not ‘full’ patterns as 60 – 20 – 20 is also possible. A ‘required’ pattern is a pattern where cut pieces do not exceed quantity ordered. In the patterns 20 – 20 – 20, 20 – 20 – 20 – 20 and 20 – 20 – 20 – 20 – 20 order length 20 is cut more than twice, but it is only required twice. These patterns will not be generated. Generating only ‘full’ and ‘required’ patterns will give a huge reduction of total patterns, keeping in mind that demands are low.
In Section 4.5.2, a total of 33 patterns were generated for example 2. While generating only ‘full’ and ‘required’ patterns only following 8 patterns will be generated, which is a reduction of more than 75 %:

\[
\begin{align*}
70 & \quad 20 \\
70 & \quad 15 \quad 15 \\
60 & \quad 20 \quad 20 \\
60 & \quad 20 \quad 15 \\
60 & \quad 15 \quad 15 \\
20 & \quad 20 \quad 15 \quad 15 \\
20 & \quad 15 \quad 15 \\
15 & \quad 15 \\
\end{align*}
\]

The procedure of the exact algorithm is as follows:

1. Generating all ‘full’ and ‘required’ patterns \(f (j = 1 \ldots n)\)
2. Adding the columns \(a_{ij} (i = 1 \ldots m)\) to the master problem
3. Solving the ‘integer’ master problem

\[
\begin{align*}
\text{Min} & \quad \sum_{j}^{n} x_{j} \\
\text{Subject to} & \quad \sum_{j}^{n} a_{ij} x_{j} \geq d_{i} \quad \forall i \quad \pi_{i} \\
x_{j} & \geq 0 \text{ and int} \quad \forall j
\end{align*}
\]

The first step and the building of the master problem will be explained thoroughly in Chapter 5, while developing the Visual Basic code for the exact algorithm.

The low demand column generation algorithm will be more suitable for the future. As seen in previous section, nowadays only three veranda project can be combined. The exact algorithm will solve these little problems fast enough and the optimal solution is ensured. When the planned purchase of the automatic sawing machine and labeller will be done, the company will be able to combine a lot of orders. This will give large problems with low demands, which will be solved very fast with the improvement of Gilmore and Gomory’s algorithm.
Chapter 5

Company Software

The next step is making an easy-to-use program, which is compatible with the company’s system by using Visual Basic. In this chapter the complete Visual Basic code will be discussed. First, we will give an introduction of \textit{lp	extunderscore solve}. As the AMPL model is using the \textit{cplex} solver, we will use \textit{lp	extunderscore solve} in our Visual Basic (VB) model. Subsequently, the low demand column generation algorithm will be programmed and explained step by step. Next, we will write some code to eliminate the excess of cut pieces. The running of the program will be illustrated with some little examples. Thereupon, the VB-code for the exact algorithm will be developed. Finally, in order to see its benefit, the results will be compared with the situation without the optimizing program using real data of the company. The complete VB-code of both programs can also be consulted in Appendices C and E.

5.1 Introduction to \textit{lp	extunderscore solve}

Programming a cutting stock problem in Visual Basic is no sinecure. First of all, we have the limitations of the Excel solver. This solver can handle only a maximum of 200 variables. As the code is not public, it is also not nice to program with it and for sure not practical. The Excel solver is also slow and not powerful.

One of the best linear programming solvers free available on the internet is \textit{lp	extunderscore solve}. \textit{Lp	extunderscore solve} is an open-source Mixed Integer Linear Programming (MILP) solver based on the revised simplex method and the Branch-and-Bound method for the integers. It was originally developed by Michel Berkelaar at Eindhoven University of Technology. It can be called as a library from different languages like C, C++, Pascal, Delphi, Java, VB, C#, VB.NET, Excel...
Any programming language capable of calling external libraries (DLLs under Windows) can call \textit{lp\_solve}.

Much more information about \textit{lp\_solve} can be found on the following web page: \texttt{http://lpsolve.sourceforge.net/5.5/}. To download the source, the libraries and manuals, membership of the \textit{lp\_solve} community is needed. The community can be accessed via the Yahoo group \texttt{http://groups.yahoo.com/group/lp\_solve/}.

### 5.2 VB-code for low demand column generation algorithm

In this section the Visual Basic code for the low demand column generation algorithm will be developed and explained step by step: accessing the \textit{lp\_solve} library, reading the input data from Excel, building up of the master problem and knapsack problem, running the algorithm and writing the solution to Excel. The complete VB-code can be found in Appendix C.

#### 5.2.1 Adaptation of the algorithm

For simplifying the programming of the algorithm in Visual Basic, the knapsack problem was adapted a little bit.

\[
\begin{align*}
\text{Max} & \quad \sum_{i}^{m} \pi_{i} a_{ij} \\
\text{Subject to} & \quad \sum_{i}^{m} a_{ij} l_{i} \leq L \\
& \quad 0 \leq a_{ij} \leq d_{i} \quad \text{and} \quad \text{int} \quad \forall \ i
\end{align*}
\]

Instead of a minimization problem with a minus sign before all the variables and a constant one in the objective function we now get a maximization problem with a positive sign before all the variables and no constant. The loop will now be stopped when the objective function of the knapsack problem is equal or smaller than one, instead of equal or greater than zero in the algorithm presented in Chapter 4.
5.2.2 Library call and declaration

Option Explicit

Public lpsolve As lpsolve55
Public Sub Cutting_Stock_Problem()
    Cleanup
    Set lpsolve = New lpsolve55
    Debug.Print CurDir$
    lpsolve.Init Application.ActiveWorkbook.Path

    ..........rest of the program.........

    Set lpsolve = Nothing
End Sub

In this first part of the program the external library lpsolve55.dll is called. Almost the full VB code is written in the Public Sub Cutting_Stock_Problem, this to make the code easier to understand and a little bit shorter. Only one other small Sub, named Cleanup, is made. On the fourth line of the program, this Sub Cleanup is called; it cleans the desired cells in Excel, filled in by a previous run of the program. To start the program a start button was placed on the Excel Sheet. This button makes the program easier to run and comfortable for users without VB knowledge; otherwise, the program must be run from the Visual Basic editor. To make this start button working, the following code was written in Sheet 1 of the Excel objects:

Private Sub CommandButton1_Click()
    Cutting_Stock_Problem
End Sub

When the start button on the Excel sheet is hit, the Sub Cutting_Stock_Problem, which contains the whole program, is run. As seen above the first thing done by the program is cleaning up all previous filled in cells in Excel, by calling the Sub Cleanup, which contains following code:

Sub Cleanup()
    Range("G2:G1000").Select
    Range(Selection, Selection.End(xlToRight)).Select
    Selection.ClearContents
    Range("G2").Select
End Sub
We now make some general declarations: \( i, j, k \) will be used a lot as counters in loops, later in the program.

```vbnet
Dim i As Long
Dim j As Long
Dim k As Long
```

### 5.2.3 Reading input data from Excel

```vbnet
Dim Profile_Length As Integer
Profile_Length = Application.Sheets("Blad1").Cells(2, "C")
```

The total length of the profile (\( L \) in the algorithm) is read from the Excel sheet and written in the parameter \( \text{Profile\_Length} \). We note that the program was developed by a Dutch version of Excel, so \( \text{Blad1} \) is given instead of Sheet1.

```vbnet
Dim Order_Lengths As Integer
Dim Array_Order_Length() As Double
Order_Lengths = Application.Sheets("Blad1").Cells(4, "B")
ReDim Array_Order_Length(Order_Lengths)
For i = 1 To Order_Lengths
    Array_Order_Length(i) = Application.Sheets("Blad1").Cells(6 + i - 1, "B")
Next
```

Subsequently, the order lengths (\( l_i \) in the algorithm) are read from the Excel sheet and written in the array \( \text{Array\_Order\_Length} \). The length of this array is the total number of order lengths (\( m \) in the algorithm); this number is also read from Excel and written in the parameter \( \text{Order\_Lengths} \).

```vbnet
Dim Order_Demands As Integer
Dim Array_Order_Demand() As Integer
Order_Demands = Application.Sheets("Blad1").Cells(4, "B")
ReDim Array_Order_Demand(Order_Demands)
For i = 1 To Order_Demands
    Array_Order_Demand(i) = Application.Sheets("Blad1").Cells(6 + i - 1, "C")
Next
```
The same thing as for the order length is done for the quantity ordered \((d_i)\) in the algorithm, the data is written in the array \(Array\_Order\_Demand\), which has of course the same length as the array \(Array\_Order\_Length\).

### 5.2.4 Defining master problem

In this part we build up the master problem. Several special \(lp\_solve\) functions from the library will be used. These functions can be recognized by a point before the function name.

```vbnet
Dim Array_Row_Zero() As Double
ReDim Array_Row_Zero(Order_Lengths)
For i = 1 To Order_Lengths
    Array_Row_Zero(i) = 1
Next

First we make an array with all the elements equal to 1, this will be used to input the coefficients of \(x_j\) in the objective function of the master problem. These coefficients are for the master problem all equal to 1:

\[
Min \sum_{j} 1 \cdot x_j
\]

```vbnet
Dim Master_Problem As Long
With lpsolve
    Master_Problem = .make_lp(0, Order_Lengths)
    .set_outputfile Master_Problem, Application.ActiveWorkbook.Path & "\Master_Problem.txt"
    .set_add_rowmode Master_Problem, True
    .set_obj_fn Master_Problem, Array_Row_Zero(0)
    Dim ColNo As Long
    Dim SparseRow As Double
    ColNo = 1
    For i = 1 To Order_Lengths
        SparseRow = Min(Int(Profile_Length / Array_Order_Length(i)), Array_Order_Demand(i))
        .add_constraintex Master_Problem, 1, SparseRow, ColNo, GE, Array_Order_Demand(i)
        ColNo = ColNo + 1
    Next
    .set_add_rowmode Master_Problem, False
End With

‘With lpsolve’ must be written to use function from the library. The master problem is built up with the function \(\text{make}\_lp\) \((\text{int rows}, \text{int columns})\), \text{rows} contains the initial number of rows and \text{columns} the initial number of columns. The \(\text{make}\_lp\) function constructs a new LP, and
sets all variables to initial values. The matrix contains no values, but space for one value. All arrays that depend on rows and columns are allocated. We start with 0 rows and \( m \) columns (total number of order lengths). The rows, which are the constraints of the master problem, will be added afterwards.

Next, the output file is set by following function: \( \text{set_outputfile}(\text{long } lp, \text{ string } filename) \), \( lp \) is a pointer to previously created lp model, \( filename \) is the file where the results will be printed to. The \( \text{set_outputfile} \) function defines the output file when \( lp\_solve \) has something to report. This output file is used for debugging purposes and will contain all the new created patterns, the value of the objective function, the number of iterations and the total time to find the solution, the maximum Branch-and-Bound level… An example of this output file can be consulted in Appendix D.

Subsequently, the rows are added. To accelerate the algorithm the \( \text{add_row_mode} \) is set by the function: \( \text{set_add_rowmode}(\text{long } lp, \text{ TRUE or FALSE}) \). A LP can be built by adding rows or columns; we will now build the linear program by adding rows. Further in the algorithm, the newly generated pattern by solving the knapsack problem will be added by adding a column. If the \( \text{rowmode} \) is FALSE, then adding columns performs better, if TRUE, then adding constraints (or rows) performs better. This is due to the construction of the \( lp\_solve \) code. Default the rowmode is FALSE.

First we set the coefficients of the objective function. This is done by the function

\( \text{set_obj_fn}(\text{long } lp, \text{ double } row) \), \( row \) is an array with \( m + 1 \) elements that contains the coefficients of \( x_j \) in the objective function. We note that element 0 of the array is not considered (i.e. ignored), column 1 is element 1, column 2 is element 2,... \( Lp\_solve \) uses element 0 of an array as a pointer and refers to all the other elements of that array. This is why in the function \( \text{set_obj_fn} \) only the element 0 is given.

We now calculate the initial patterns:

\[
a_i = \min \{ \left[ L/l_i \right] \text{ and } d_i \} \quad \text{for } i = 1 \ldots m
\]

In VB this becomes:

\[
\text{Min}(\text{Int(Profile\_Length} / \text{Array\_Order\_Length}(i)), \text{Array\_Order\_Demand}(i))\).
\]
Thereupon the constraints can be added for each order length $i$:

$$\sum_j a_{ij} x_j \geq d_i \quad \forall i$$

by using the function `add_constraintex(long lp, int count, double row, int colno, int constr_type, double rh)`:

- `count` contains the number of elements in `row` and `colon`;
- `row` is an array with $m + 1$ elements that contains the coefficients of $x_j$ in the objective function;
- `colon` is a zero-based array with `count` elements that contains the column numbers of the row;
- `constr_type` set the type of the constraint (LE = Less than or equal ($\leq$); EQ = Equal ($=$); GE = Greater than or equal ($\geq$));
- `rh` contains the value of the right hand side (RHS) ($d_i$ in the algorithm).

The `add_constraintex` function adds a row to the model (at the end) and sets all values of the row at once. We note that element 0 of the array is not considered for `add_constraintex` (i.e. ignored), column 1 is element 1, column 2 is element 2,...

### 5.2.5 Defining knapsack problem

As we build the master problem in the previous part, we will now build the knapsack problem, which is another optimization problem.

```vbnet
Dim Knapsack As Long
With lpsolve
    Knapsack = .make_lp(0, Order Lengths)
    .set_outputfile Knapsack, Application.ActiveWorkbook.Path & "\Knapsack.txt"
    .set_maxim Knapsack
    .set_add_rowmode Knapsack, True
        .add_constraint Knapsack, Array_Order_Length(0), LE, Profile Length
    For i = 1 To Order Lengths
        .set_upbo Knapsack, i, Array_Order_Demand(i)
    Next
    .set_add_rowmode Knapsack, False
    For i = 1 To Order Lengths
        .set_int Knapsack, i, True
    Next
End With
```
As for the master problem, we use the functions `make_lp` and `set_outputfile` to make the linear program and set the output file. When constructing a new LP with `make_lp`, the problem is set by default as a minimization problem. Since the knapsack problem is a maximization problem, we have to change this. This can be done by the function `set_maxim(long lp)`.

We will again build the LP by adding rows, so the `set_add_rowmode` function is used. The knapsack problem contains only one constraint:

$$\sum_{i}^{m} a_{ij} l_j \leq L.$$  

This constraint is added by the function `add_constraint(long lp, double row, int constr_type, double rh)`, which is analogue to the function `add_constraintex` used for constructing the master problem. The difference is that with the last function only the non-zero elements are given. This speeds up the model building considerably if there are a lot of zero values, like for the master problem. As in the knapsack constraint all coefficients are different from zero, we use the `add_constraint` function.

Subsequently, we set the upper bound of the variables ($d_i$ in the algorithm) by using the function `set_upbo(long lp, int column, double value)`, `column` contains the column number of the variable on which the bound must be set. The `set_upbo` function sets an upper bound on the variable identified by `column`. Setting a bound on a variable does not increase the model size, as adding an extra constraint (row) to the model. This means that the model stays smaller and will be solved faster. The default upper bound of a variable is infinity (in fact a very big number).

Finally, we make the variables integer, as by default a variable is not integer. This can be done by following function: `set_int(long lp, int column)`, `column` contains the column number of the variable that must be set.

We note that the objective function has not been inputted, as this will be done later when the shadow prices of the master problem are known.
5.2.6 The Cutting Stock Algorithm

We now have constructed the master and knapsack problem and can program the loop of algorithm, steps 2 to 4.

```vbnet
Dim Obj As Double
Dim Flag As Boolean

Dim Array_New_Column() As Double
ReDim Array_New_Column(Order_Lengths)

Dim Array_Duals() As Double

With lpsolve
    Obj = 10
    Flag = False
    While Obj > 1.00001
        If Flag = True Then
            ReDim Array_New_Column(1 To .get_Ncolumns(Knapsack) + 1)
            .get_variables Knapsack, Array_New_Column(1)
            For i = Order_Lengths + 1 To 2 Step -1
                Array_New_Column(i) = Array_New_Column(i - 1)
            Next
            Array_New_Column(1) = 1
            .add_column Master_Problem, Array_New_Column(1)
        End If
        .solve Master_Problem
        ReDim Array_Duals(0 To .get_Ncolumns(Master_Problem) +
                            .get_Nrows(Master_Problem))
        .get_dual_solution Master_Problem, Array_Duals(0)
        .set_add_rowmode Knapsack, True
        .set_obj_fn Knapsack, Array_Duals(0)
        .set_add_rowmode Knapsack, False
        .solve Knapsack
        Obj = .get_objective(Knapsack)
        Flag = True
    Wend
End With
```

The loop will be stopped, when the objective function of the knapsack problem is smaller than or equal to one. In VB a while-loop was constructed that runs as long as the objective function of the knapsack problem is greater than one. We note that 1,00001 was used instead of 1, to make the program stable.
In the last loop, when the objective function of the knapsack problem becomes equal or smaller than one, no new pattern is added to the master problem. Therefore the part of the code which adds a new column to the master problem is placed before the rest of the code. The Boolean flag ensures that this part of the code is not run in the first loop.

First the master problem is solved by using the function \texttt{solve(long }lp\texttt{)}; this is step 2 of the algorithm. Next, the shadow prices or dual variables are saved in the array \texttt{Array\_Duals}; this can be done by using the following function: \texttt{get\_dual\_solution(long }lp\texttt{, double }duals\texttt{)}, \texttt{duals} is an array that will contain the values of the dual variables. The \texttt{get\_dual\_solution} function returns the values of the dual variables and needs an array that is already dimensioned with \texttt{l + number of rows (m) + number of columns (n)} elements. We note that the index starts from 1 and element 0 is not used.

To get the number of rows (constraints) and the number of columns (variables) of the master problem the functions \texttt{get\_Nrows(long }lp\texttt{)} and \texttt{get\_Ncolumns(long }lp\texttt{)} are used. The rows will always be equal to the number of order lengths (\texttt{m}), the number of columns depends on the number of new patterns that have been found.

Once the shadow prices are known, we can set the objective function of the knapsack problem:

\[ \text{Max} \quad \sum_{i}^{m} \pi_{i} a_{ij} \]

by using the \texttt{set\_obj\_fn} function. The row mode is again set on to do this. Now the knapsack problem can be solved, step 3 of the algorithm. The value of the objective function is stored in the integer \texttt{Obj}. For this purpose the function \texttt{get\_objective(long }lp\texttt{)} is used.

If the objective function is greater than one, step 4 of the algorithm, the variables are added to the master problem. The variables of the knapsack problem give the new pattern or column in the master problem. As mentioned above, the code to add a new column to the master problem was placed before the rest of the code. We can retrieve the variables with the function \texttt{get\_variables(long }lp\texttt{, double }var\texttt{)}; \texttt{var} is an array that will contain the values of the variables. The function \texttt{get\_variables} needs an array that is already dimensioned with \texttt{m} elements or the number of columns of the knapsack problem. The variables are stored in the array \texttt{Array\_New\_Column}. This array was dimensioned starting from 1 (instead of zero) to \texttt{m}. 

100
+ 1, and contains \( m + 1 \) elements, so one more than necessary. Element 1 will contain the value of the first variable, element 2 of the second variable, ... Now all elements of that array will be shifted one position further. This is the reason why we dimensioned it with one more element than necessary! Subsequently, in the first element of the array, the coefficient of the objective function of the master problem is stored, this element is always one.

We give a little example to elucidate this. Suppose the variables of the objective function are \([2 \ 1 \ 0 \ 1]\). The array \( \text{Array\_New\_Column} \) will be \([2 \ 1 \ 0 \ 1 \ 0]\), but after shifting we get \([0 \ 2 \ 1 \ 0 \ 1]\). Finally, we set the first element of the array \([1 \ 2 \ 1 \ 0 \ 1]\). This array will be used to add a column to the master problem.

As commented above, in \( \text{lp\_solve} \) the element 0 of an array is used as a pointer for all the other elements. In the previous part of the program the element 0 was always used to pass on the data. Since we want to use the \( \text{Array\_New\_Column} \) to add a new column to the master problem and the first element of that column must contain the coefficient of the objective function (row 0) we need to store some data in the first element. In a normal array the first element has index equal to zero and storing data in the first element would not be possible. To skirt this difficulty we used a little trick, dimensioning the array starting from index 1!

We now will add the array \( \text{Array\_New\_Column} \) to the master problem by using the function \( \text{add\_column(long \ lp, double \ column)} \), \( \text{column} \) is an array with \( m + 1 \) elements that contains the values of the column. We note that for \( \text{add\_column} \) element 1 of the array is the value of the objective function for that column, element 2 will contain the first element of the pattern, element 3 will contain the second element of the pattern, ...

```plaintext
With lpsolve
    For i = 0 To .get_Ncolumns(Master_Problem)
        .set_int Master_Problem, i, True
    Next
    .solve Master_Problem
End With
```

Now the objective function of the knapsack is 1 (or very close to 1), so no new patterns will be found to improve the relaxed solution of the master problem and we have an optimal non-integer solution. In step 5 of the algorithm the master problem is solved a last time, but now with integer variables. With the function \( \text{set\_int} \) the variables are set to integer and thereupon
the master problem is solved. This is the end of the algorithm and we now have to interpret
the solution.

5.2.7 Writing the solution to Excel

In this section of the program the solution is extracted to Excel and the waste is calculated.

```vbnet
Dim Array_Cut() As Double
Dim Array_Pattern() As Double
Dim used_length As Integer
Dim waste As Integer
Dim percentage_waste As Double
Dim CutNo As Integer
CutNo = 0
With lpsolve
    ReDim Array_Cut(1 To .get_Ncolumns(master_problem))
    .get_variables master_problem, Array_Cut(1)
    First the variables of the ‘integer solved’ master problem are stored in the array Array_Cut;
    the function get_variables does this.
    For j = 1 To .get_Ncolumns(master_problem)
        If Array_Cut(j) <> 0 Then
            ReDim Array_Pattern(.get_Nrows(master_problem))
            .get_column master_problem, j, Array_Pattern(0)
            CutNo = CutNo + 1
            Application.Sheets("Blad1").Cells(4, Chr(65 + 5 + CutNo)) = Array_Cut(j)
            Application.Sheets("Blad1").Cells(5, Chr(65 + 5 + CutNo)) = "Pattern " & CutNo
            For i = 1 To Order_lengths
                Application.Sheets("Blad1").Cells(5 + i, Chr(65 + 5 + CutNo)) = Array_Pattern(i)
            Next
            used_length = 0
            For i = 1 To Order_lengths
                used_length = used_length + (Array_Pattern(i) * Array_Order_length(i))
            Next
            waste = waste + (Array_Cut(j) * (Profile_length - used_length))
        End If
    Next
End With
```

Subsequently, all the patterns (columns of the master problem) that are used in the final solution are written in Excel. We make a loop, which looks if the variable of the master problem ($x_j$ in the algorithm) defers from zero, for all columns. If $Array_Cut (j) \neq 0$, then the variable of the master problem is not 0, and the pattern is not used; so, it will not be written to Excel.

If the pattern is used, all values of the column are retrieved by using the function

$\text{get\_column(long } lp, \text{ int } col\_nr, \text{ double } column)\text{, } col\_nr \text{ contains the column number of the column to retrieve, } column \text{ is an array in which the values are returned. The function }$ $\text{get\_column} \text{ needs an array that must be dimensioned with at least } m + 1 \text{ elements. Element 0 of the } column \text{ array will contain the variable (number of times the pattern is used), element 1 will contain the first element of the pattern, element 2 will contain the second element of the pattern,… The pattern is stored in the array } Array\_Pattern.$

Now all the information to write the following data to Excel has been collected: (impersonal!)

- the number of times the pattern is used ($Array_Cut (j)$ in the VB-code),
- the name of the pattern, starting with pattern 1, …,
- the elements of the pattern.

Thereupon some code is written to calculate the waste. First, for each pattern $j$ the used length is calculated: $\sum_{i=1}^{m} a_{ij} \cdot l_i$. In VB this becomes:

$$\text{used\_length} = \text{used\_length} + (\text{Array\_Pattern}(i) * \text{Array\_Order\_length}(i)).$$

Next, the waste can be calculated: $\sum_{j=1}^{n} x_j \cdot (L - \text{used\_length}_j)$. In VB it is:

$$\text{waste} = \text{waste} + (\text{Array\_Cut}(j) * (\text{Profile\_length} - \text{used\_length})).$$

The waste is stored in the integer $\text{waste}$.

$$\frac{\text{percentage\_waste}}{100} = \frac{\text{waste}}{(\text{Profile\_length} * \text{.get\_objective(master\_problem)})} \times 100$$

$\text{Application.Sheets("Blad1").Cells(2, "G") =}$

$\text{.get\_objective(master\_problem)}$

$\text{Application.Sheets("Blad1").Cells(3, "G") = waste}$

$\text{Application.Sheets("Blad1").Cells(3, "H") = Round(percentage\_waste, 3)}$

$\text{/ 100}$

$\text{.print\_lp master\_problem}$

$\text{.print\_lp knapsack}$
In the last part of the program, the waste is calculated in terms of percentage:

\[
\% \text{ waste} = \frac{\text{waste}}{L \times \text{total number of profiles used}} \times 100.
\]

Subsequently, the total number of profiles used (value of the objective function in the master problem), the waste and the percentage of waste are written to Excel.

The function `print_lp(long lp)` prints the linear programming model to the output file which was set before, used for debugging purposes.

### 5.2.8 Eliminating excess of cut pieces

In Section 4.7.4, while running the AMPL program, we have seen that sometimes some pieces are cut too many times. This is a logic consequence of the algorithm as nowhere is said that the total pieces cut must be equal to the demand. The constraints in the master problem say that the demand must be fulfilled, using an equal or greater sign. In some cases the demand will be more than fulfilled as the combination of the best patterns found by solving the knapsack problem results in a surplus of a certain order length.

We will now adapt and replace the last part of the program ‘writing the solution to Excel’ to eliminate the excess of cut pieces. Instead of working with the one-dimensional array `Array_Pattern`, we will now work with the two-dimensional array `Array_Patterns`. This is necessary as we have to store all the patterns in the memory. In Section 5.2.7 the pattern was stored in the array `Array_Pattern` and overwritten every time a new used pattern was found. Working with a two-dimensional array is a first thing that will enlarge the program code.

```vba
Dim Array_Cut() As Double
Dim Array_Patterns() As Double
Dim Array_Quantity_Cut() As Double
Dim ProfileNo As Integer
Dim ProfileNoBis As Integer
```
Dim Used_Length As Integer
Dim Waste As Integer
Dim Percentage_Waste As Double

With lpsolve
    ReDim Array_Cut(1 To .get_Ncolumns(Master_Problem))
    .get_variables Master_Problem, Array_Cut(1)
    ReDim Array_Patterns(.get_Nrows(Master_Problem),
                        .get_objective(Master_Problem))

    For j = 1 To .get_Ncolumns(Master_Problem)
        If Array_Cut(j) <> 0 Then
            For k = Array_Cut(j) To 1 Step -1
                ProfileNo = ProfileNo + 1
                .get_column Master_Problem, j, Array_Patterns(0, ProfileNo)
            Next
        End If
    Next
End With

As in above section all the columns (patterns) of the master problem are analysed and if the variable \((x_j)\) in the algorithm) defers from zero, the pattern is stored. The functions \textit{get_column}, \textit{get_variables} and \textit{get_objective} are used. The first difference is that now the pattern will not be written immediately to Excel. We do not know yet if there are some excessive cuts in the pattern or not. Secondly the patterns are, as already mentioned, in a two-dimensional array \textit{Array_Patterns}. A third difference is that the patterns are split up; a pattern used three times in the final solution will be stored three times in the array \textit{Array_Patterns}. This is done by the k-loop and is necessary to eliminate the excess of cut pieces in further part of the code. Because of this, the number of patterns will not be counted, only the number of profiles. The total number of profiles counted will always be equal to the value of the objective function of the master problem.

ReDim Array_Quantity_Cut(Order_Lengths)

For i = 1 To Order_Lengths
    For j = 1 To ProfileNo
        Array_Quantity_Cut(i) = Array_Quantity_Cut(i) +
        Array_Patterns(i, j)
    Next
Next

Thereupon, the total pieces cut for each order length are counted and stored in the array \textit{Array_Quantity_Cut}. We analyse every profile (stored in \textit{Array_Patterns}) and add up all the pieces cut of that particular order length.
Next, for each order length we compare the array `Array.Quantity_Cut` and the array `Array.Order_Demand`. If the array `Array.Quantity_Cut` is higher than the array `Array.Order_Demand`, there is an excess of pieces cut for that particular order length. We note the first array can never have smaller values than the second array, because of the sign of the master problem constraints. For every piece that is cut too much, one piece is taken out of the profiles used; this is done by the k-loop. The j-loop analyzes, for every profile, if a piece of that particular order length is cut. If it is true, the excessive piece is removed and the flag is put on true, so the j-loop is stopped and the k-loop continues.

```vbnet
For i = 1 To Order_Lengths
    If Array.Quantity_Cut(i) > Array.Order_Demand(i) Then
        For k = Array.Quantity_Cut(i) To Array.Order_Demand(i) + 1 Step -1
            Flag = False
            For j = 1 To ProfileNo
                If Array.Patterns(i, j) > 0.01 And Flag = False Then
                    Array.Patterns(i, j) = Array.Patterns(i, j) - 1
                    Flag = True
                End If
            Next
        Next
    End If
Next
```

```
Next, for each order length we compare the array `Array.Quantity_Cut` and the array `Array.Order_Demand`. If the array `Array.Quantity_Cut` is higher than the array `Array.Order_Demand`, there is an excess of pieces cut for that particular order length. We note the first array can never have smaller values than the second array, because of the sign of the master problem constraints. For every piece that is cut too much, one piece is taken out of the profiles used; this is done by the k-loop. The j-loop analyzes, for every profile, if a piece of that particular order length is cut. If it is true, the excessive piece is removed and the flag is put on true, so the j-loop is stopped and the k-loop continues.

```vbnet
For j = 1 To ProfileNo
    Flag = False
    For i = 1 To Order_Lengths
        If Array.Patterns(i, j) <> Array.Patterns(i, j - 1) And Flag = False Then
            Flag = True
            ProfileNoBis = ProfileNoBis + 1
            k = 0
        End If
    Next
    If Flag = True Then
        Application.Sheets("Blad1").Cells(4, Chr(65 + 5 + ProfileNoBis)) = 1
        Application.Sheets("Blad1").Cells(5, Chr(65 + 5 + ProfileNoBis)) = ”Pattern ” & ProfileNoBis
        Used_Length = 0
        For i = 1 To Order_Lengths
            Application.Sheets("Blad1").Cells(5 + i, Chr(65 + 5 + ProfileNoBis)) = Array.Patterns(i, j)
            Used_Length = Used_Length + (Array.Patterns(i, j) * Array.Order_Length(i))
        Next
        Waste = Waste + (Profile_Length - Used_Length)
    Else
        Application.Sheets("Blad1").Cells(4, Chr(65 + 5 + ProfileNoBis)) = 1
        Application.Sheets("Blad1").Cells(5, Chr(65 + 5 + ProfileNoBis)) = ”Pattern ” & ProfileNoBis
        Used_Length = 0
        For i = 1 To Order_Lengths
            Application.Sheets("Blad1").Cells(5 + i, Chr(65 + 5 + ProfileNoBis)) = Array.Patterns(i, j)
```
We now have all the necessary profiles stored in the two-dimensional array $Array\_Patterns$, without any excessive cut piece. The last thing to do is joining the identical profiles. A profile that is used four times would otherwise be written four times to Excel. We get a better looking result when that profile is written only once with the mention that it has to be used four times (multiplication factor four). The above VB-code to join the profiles and write the multiplication factor is complex. We note that, if some pieces were cut too often, the number of different profiles used will now be greater than with the code of Section 5.2.7. Therefore, $Profile\_No\_Bis$ is used in the VB-code instead of $Profile\_No$. Some patterns were changed to equalize the pieces cut and the quantity ordered for each order length. The total number of profiles used will always be, of course, the same given by the objective value of the master problem.

For every profile (j-loop) we analyze if it is the same as the previous profile (i-loop). If it is, the flag is set on true, the counter $Profile\_No\_Bis$ is increased by one and $k$ is set zero. The integer $k$ is a second counter that will count how many times the profile is used.

If the flag is true, the profile is written to Excel, the multiplication factor is set to one and the waste is calculated. This part of the code is analogue to the corresponding code in Section 5.2.7. The differences are: a two-dimensional array $Array\_Patterns$ and another counter $Profile\_No\_Bis$.

If the flag is false, the multiplication factor is changed according to $k$. For example if $k = 1$, the flag was two consecutive times false, which means the profile was two times equal to the previous profile. The multiplication factor will then be set to $2 + k = 3$, which is correct. Also the waste is added up. It is not necessary to calculate the used length as an already used profile is being used.

\[
\text{Percentage\_Waste} = \left( \frac{\text{Waste}}{(\text{Profile\_Length} \times \text{.get\_objective(Master\_Problem)})} \right) \times 100
\]

Application.Sheets("Blad1").Cells(2, "G") =
\text{.get\_objective(Master\_Problem)}
Application.Sheets("Blad1").Cells(3, "G") = Waste
Application.Sheets("Blad1").Cells(3, "H") = Round(Percentage_Waste, 3) / 100
.print_lp Master_Problem
.print_lp Knapsack
End With

Set lpsolve = Nothing
End Sub

In the last part of the code the percentage of waste is calculated, and the total number of profiles used, the waste and the percentage of waste are written to Excel. This part of the code is equal to that in Section 5.2.7, so no further clarification is needed.

### 5.2.9 Illustration of Visual Basic program

In this section we will illustrate the developed Visual Basic program by solving two previously used examples with the program and display the results.

In Section 4.6.1 example 1 was used, to show that Gilmore and Gomory’s algorithm does not work when the demands are low. Order Length and Quantity Ordered are:

<table>
<thead>
<tr>
<th>Order Length</th>
<th>Quantity Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table 5-1  Repetition of example 1 CSP**

We repeat that in Section 4.6.1, with Gilmore and Gomory’s algorithm, the following solution was found:

- 50 – 50
- 20 – 20 – 20 – 20 – 20
- 30 – 30 – 30 – 10

In total five stock profiles are used.
With the Visual Basic program we get the result presented below in table 5-2. The yellow fields can be filled in, and will be used as data in the program: the total length of the stock profiles, the different order lengths and the respectively quantity ordered. We note that there are only seven yellow fields to fill in the order length and quantity ordered. This is only done for the presentation and in practice the number of different order lengths and quantity ordered are unlimited. The other (non-yellow) fields are automatically generated after a click on the start button.

Table 5-2 shows the output before a click on the solve button and Table 5-3 shows the output after it. Instead of five stock profiles, with Gilmore and Gomory’s algorithm we only need two stock profiles, which is the optimal solution:

50 – 25 – 25  
30 – 20 – 20 – 10 – 10 – 10
### Table 5-2  Output before a click on the solve button

<table>
<thead>
<tr>
<th>Profile Length</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>5</td>
</tr>
<tr>
<td>Order Length</td>
<td>Quantity Ordered</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

### Table 5-3  Output after a click on the solve button

<table>
<thead>
<tr>
<th>Profile Length</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>5</td>
</tr>
<tr>
<td>Order Length</td>
<td>Quantity Ordered</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

| Total profiles | 2   |
| Waste          | 0   |
| Quantity       | 1   |
| Pattern 1      | 1   |
| Pattern 2      | 2   |
| Order Length   | 50  |
|                | 30  |
|                | 25  |
|                | 20  |
|                | 10  |
We will now solve Example 3, as was done with the AMPL program. This example was made from real data that can be found in Section 4.4 for the profile VER11/10. Order Length and Quantity Ordered are:

<table>
<thead>
<tr>
<th>Order Length</th>
<th>Quantity Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1868</td>
<td>2</td>
</tr>
<tr>
<td>275</td>
<td>2</td>
</tr>
<tr>
<td>1878</td>
<td>2</td>
</tr>
<tr>
<td>287</td>
<td>2</td>
</tr>
<tr>
<td>2874</td>
<td>2</td>
</tr>
<tr>
<td>2858</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5-4 Example 3 CSP

The results obtained with the Visual Basic program are presented below. Table 5-6 displays the result without removing the excessive pieces cut. Table 5-7 displays the result where the last part of the VB-code is adapted as explained in Section 5.4. We observe that the waste is bigger in the second solution. This is logic as in the first solution a part of the waste is filled with no needed pieces.

Another example to illustrate the deletion of the excessive pieces is following:

<table>
<thead>
<tr>
<th>Order Length</th>
<th>Quantity Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>2</td>
</tr>
<tr>
<td>60</td>
<td>3</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5-5 Example 4 CSP

Without removing the excess pieces, order length 20 is cut two times in pattern 3. But the pattern is used two times, so order length 20 is cut four times in total. Order length 15 is cut two times in pattern 4 and two times in pattern 5, so also four times (see table 5-8). Both order lengths are only needed three times. With the adaptation of the last part of the VB program we obtain a good result.
Table 5-6  Output of example 3 without removing the excessive pieces

<table>
<thead>
<tr>
<th>Order Length</th>
<th>Quantity Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1868</td>
<td>2</td>
</tr>
<tr>
<td>275</td>
<td>2</td>
</tr>
<tr>
<td>1878</td>
<td>2</td>
</tr>
<tr>
<td>287</td>
<td>2</td>
</tr>
<tr>
<td>2874</td>
<td>2</td>
</tr>
<tr>
<td>2858</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5-7  Output of example 3 with removing the excessive pieces

<table>
<thead>
<tr>
<th>Order Length</th>
<th>Quantity Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1868</td>
<td>2</td>
</tr>
<tr>
<td>275</td>
<td>2</td>
</tr>
<tr>
<td>1878</td>
<td>2</td>
</tr>
<tr>
<td>287</td>
<td>2</td>
</tr>
<tr>
<td>2874</td>
<td>2</td>
</tr>
<tr>
<td>2858</td>
<td>1</td>
</tr>
</tbody>
</table>
### Table 5-8  Output of example 4 without removing the excessive pieces

<table>
<thead>
<tr>
<th>Profile Length</th>
<th>Quantity</th>
<th>Total profiles</th>
<th>Waste</th>
<th>Total</th>
<th>Pattern 1</th>
<th>Pattern 2</th>
<th>Pattern 3</th>
<th>Pattern 4</th>
<th>Pattern 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>2</td>
<td>40</td>
<td>6.67%</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 5-9  Output of example 4 with removing the excessive pieces

<table>
<thead>
<tr>
<th>Profile Length</th>
<th>Quantity</th>
<th>Total profiles</th>
<th>Waste</th>
<th>Total</th>
<th>Pattern 1</th>
<th>Pattern 2</th>
<th>Pattern 3</th>
<th>Pattern 4</th>
<th>Pattern 5</th>
<th>Pattern 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>2</td>
<td>40</td>
<td>6.67%</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

---

Chapter 5

Visual Basic Program
5.3 **VB-code for exact algorithm**

In this section the VB-code for the exact algorithm will be developed. The beginning and the end of the program code are identical to the program code of the low demand delayed column generation algorithm: library call and declaration (Section 5.2.2), reading input data from Excel (Section 5.2.3) and writing solution to Excel (Section 5.2.8). First, we will describe how the exact algorithm is programmed with words. Next, a little Sub to sort the order lengths is developed. Then we will define the master problem and solve it. In the exact algorithm there is no knapsack problem to find new patterns; so, this part of the code has not to be programmed. Since the program code is not so easy to understand, after explaining it completely, an illustrative example will be given which shows the values of all counters and arrays from the beginning to the end of the program run. The complete VB-code can be consulted in Appendix E.

### 5.3.1 Generating all ‘full’ and ‘required’ patterns

The program will begin with the biggest order length and place it in the temporary pattern. This is done until the place left is smaller than the order length, so it can not be placed anymore. Next, it will go to the second biggest order length and check if it can be added to the temporary pattern. And so on until, the smallest order length is reached. When the smallest order length can not be added anymore to the temporary pattern, a ‘full’ pattern is found and it will be added as a column to the master problem. For another pattern, we remove the last piece added to the pattern. If the last piece was the smallest order length, more than one piece will be removed from the temporary pattern. After removing pieces, the place left in the temporary pattern is again bigger and other, smaller pieces can be added to the temporary pattern. Every time the smallest order length can not be added anymore, a ‘full’ pattern is found and added to the master problem. This goes on until a pattern where all pieces are the smallest order length is added to the master problem.

When adding an order length to the temporary pattern not only the place left has to be examined, also the quantity ordered for this particular order length. Since only ‘required’
patterns will be considered, an order length will never be placed more in a pattern than it is needed.

### 5.3.2 Sorting all order lengths

To begin with the biggest and end with the smallest order length, first all order lengths have to be arranged from big to small. The following Sub sorts all the order lengths from big to small. The Sub `Sort_Order_Lengths` is placed after the Sub `Cleanup`, considered in Section 5.2.2, at the beginning of the program.

```vba
Sub Sort_Order_Lengths()
    Range("B5:C20").Sort Key1:=Range("B5"), Order1:=xlDescending, Header:=xlGuess, OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom, _ DataOption1:=xlSortNormal
End Sub
```

### 5.3.3 Defining master problem

In this part of the program, the master problem will be built and all ‘full’ and ‘required’ patterns will be added to it. Subsequently, the problem will be made integer and the right hand side, i.e. the quantity ordered will be inputted.

```vba
Dim r As Long
Dim n As Integer
Dim Array_New_Column() As Double
Dim Array_Teller() As Double
Dim Array_Pattern_Big() As Double
Dim Array_Pattern_Small() As Double
Dim Flag As Boolean
```

The letter ‘r’ will be used for the place left in the temporary pattern. At the beginning this is equal to the profile length. The letter ‘n’ will be used to indicate the reached entry place in the temporary pattern. For example, a temporary pattern 60 – 20 – 20 will have n = 4, since three pieces are already placed and the next piece would be placed on position four.

A lot of arrays are used in the VB-code. The `Array_Teller` contains the reached level of the different entry places. Level 1 corresponds with the biggest order length and so on. We consider for example a problem with only three order lengths: 60, 20 and 15. When the
Chapter 5

Visual Basic Program

temporary pattern 60 – 20 – 20 is reached, the Array_Teller will be [ 1 2 2 2]. The first entry place has the biggest order length 60. The second and third entry places have length 20. A hypothetic new entry place will always have a level equal or higher than previous entry place.

The array Array_Pattern_Big and Array_Pattern_Small are in previous example respectively [ 60 20 20 ] and [ 1 2 0 ]. Both arrays give the same information on another way. The first array shows all pieces, represented by their order length, in the temporary array. The second array displays the quantity of each order length in the temporary pattern, beginning with the biggest one.

The array Array_New_Column is the column that will be add to the master problem. This column was also used in the low demand column generation program. When the array of the previous example 60 – 20 – 20 is add to the master problem, the array Array_New_Column will be [ 1 1 2 0 ]. This array is always the same as the array Array_Pattern_Small, with the difference that the first element contains the coefficient of the objective function of the master problem. This coefficient is always one, as we already have seen. All elements of the array Array_Pattern_Small are shifted one place further.

As seen in Section 5.2.4, we use the functions make_lp and set_outputfile to make the linear program and set the output file. Subsequently the arrays Array_Teller, Array_Pattern_Big and Array_Pattern_Small are re-dimensioned. The first element of the array Array_Teller is set to 1 and the entry level counter ‘n’ also.

```vbnet
Dim Master_Problem As Long
With lpsolve
    Master_Problem = .make_lp(Order_Lengths, 0)
    .set_outputfile Master_Problem, Application.ActiveWorkbook.Path & 
        "\Master_Problem.txt"
    ReDim Array_Teller(Int(Profile_Length / Array_Order_Length(Order_Lengths)) + 1)
    Array_Teller(1) = 1
    ReDim Array_Pattern_Big(Int(Profile_Length / Array_Order_Length(Order_Lengths)) + 1)
    ReDim Array_Pattern_Small(Order_Lengths)
    n = 1
```
We now consider the big while-loop, where all ‘full’ and ‘required’ patterns are added to the
master problem.

```
While n > 0
    r = Profile_Length
    For i = 1 To n - 1
        r = r - Array_Pattern_Big(i)
    Next
```

In the beginning of the loop the place left in the temporary pattern ‘r’ is calculated. The for-
loop goes only to ‘n-1’ because only the first ‘n-1’ elements of the array `Array_Pattern_Big`
are relevant. A new piece will be added on position n in the array, since the next entry place is
n.

```
If Array_Order_Length(Array_Teller(n)) <= r And
    Array_Pattern_Small(Array_Teller(n)) <
    Array_Order_Demand(Array_Teller(n)) Then
    Array_Teller(n + 1) = Array_Teller(n)
    Array_Pattern_Big(n) = Array_Order_Length(Array_Teller(n))
    Array_Pattern_Small(Array_Teller(n)) =
    Array_Pattern_Small(Array_Teller(n)) + 1
    n = n + 1
Else
```

Subsequently, the program will check if the current order length can be added to the
temporary pattern. The current order length is the order length reached so far in the while-
loop on the entry place given by ‘n’. The if-statement is a statement with two conditions.
Since `Array_Teller(n)` gives the level of current entry place,
`Array_Order_Length(Array_Teller(n))` will give the current order length. The first
condition checks if the current order length is smaller than the place over in the temporary
pattern. The second condition verifies that the quantity of the current order length in the
temporary pattern is smaller than the ordered quantity.

If these two conditions are true: the array `Array_Teller` will be increased by one, the current
order length will be added to the array `Array_Pattern_Big`, the quantity of the current order
length will be increased by one in the array `Array_Pattern_Small` and finally the entry place
‘n’ will be moved up as a piece was added. Subsequently the while-loop is re-run.
If one of these two conditions is not true, the current order length will not be added to the temporary pattern and the code below is run.

If Array_Teller(n) < Order_Lengths Then
    Array_Teller(n) = Array_Teller(n) + 1
Else
    ReDim Array_New_Column(1 To Order_Lengths + 1)
    ReDim Array_Pattern_Small(Order_Lengths)
    For i = 1 To n - 1
        Flag = False
        For j = 1 To Order_Lengths
            If Array_Pattern_Big(i) = Array_Order_Length(j) And Flag = False Then
                Array_New_Column(j + 1) = Array_New_Column(j + 1) + 1
                Flag = True
            End If
        Next
    Next
    Array_New_Column(1) = 1
    .add_column Master_Problem, Array_New_Column(1)
    k = 1
    While Array_Teller(n - k) = Order_Lengths
        k = k + 1
    Wend
    Array_Teller(n - k) = Array_Teller(n - k) + 1
    n = n - k
    For i = 1 To n - 1
        While Flag = False
            For j = 1 To Order_Lengths
                If Array_Pattern_Big(i) = Array_Order_Length(j) And Flag = False Then
                    Array_Pattern_Small(j) = Array_Pattern_Small(j) + 1
                    Flag = True
                End If
            Next
        Next
    End If
End If
Wend

In this part of the while-loop we have again an if-statement. The if-statement checks if the level of current entry place is smaller than the number of order lengths. This statement will be false when the level of current entry place equals the level of the smallest order length, or in other words when the current order length is the smallest order length. If the statement is true,
the level of the current entry place is added by one and the while-loop is re-run. If the statement is false, the level of the current entry space can not be increased by one, as the maximum level is reached, corresponding with the smallest order length. In this case we are sure a ‘full’ pattern is found. No pieces can be added anymore to the end of the temporary pattern and it will be added to the master problem.

First, the arrays Array_New_Column and Array_Pattern_Small are reset. Next, a for-loop is run, which transforms the array Array_Pattern_Big into the array Array_New_Column. The Boolean Flag makes the algorithm faster, by stopping the j-loop when a match is found. After the for-loop the first element of the array Array_New_Column is set to one and the array is add to the master problem by using the function add_column.

The following little while-loop counts how many times the smallest order length is placed at the end of the pattern. Subsequently, the entry place ‘n’ is reduced by ‘k’, i.e. one more than the quantity of the smallest order length in the pattern, and the level of that entry space is incremented by one. We consider for example again a problem with following three order lengths: 60, 20 and 15. When the pattern 60 – 20 – 15 is added to the master problem, the entry space ‘n’ is 4 and the array Array_Teller will be [ 1  2  3  3 ]. If the entry space ‘n’ is only reduced by one to 3, the counter of that entry space can not be increased by one as it has already the maximum value. The entry space has to be reduced by 2, i.e. one more than the quantity of the smallest order length in the pattern. The new entry space will be 2 and the array Array_Teller [ 1  3 ].

In the subsequent for-loop, the array Array_Pattern_Big is converted to the array Array_Pattern_Small. This is needed because, in the array Array_Pattern_Big, the last added piece(s) can be easily removed, by decreasing the entry space, but in the array Array_Pattern_Small this is not the case.

```vbnet
For i = 1 To .get_Ncolumns(Master_Problem)
    .set_int Master_Problem, i, True
Next

For i = 1 To Order_Lenghts
    .set_rh Master_Problem, i, Array_Order_Demand(i)
    .set_constr_type Master_Problem, i, GE
Next

.solve Master_Problem
```
The columns of the master problem contain now all ‘full’ and ‘required’ patterns. In the first for-loop all variables corresponding with the columns or patterns are set to integer. The function set_int is used.

In the second for-loop the right hand sides are set with the function set_rh(long lp, int row, double value), row is the row for which the RHS value must be set, value contains the value of the right hand side. Subsequently, the constrained types are set. This is done by the function set_constr_type(long lp, int row, int con_type), row is the row for which the constraint type must be set, con_type set the type of the constraint: LE = Less than or equal (<=); EQ = Equal (=); GE = Greater than or equal (>=). Finally, the master problem is solved and the solution is written to Excel.

### 5.3.4 Illustrative example

In this Section the while-loop will be run, showing the value of all counters and arrays from the beginning to the end of the loop. This will give a clear view on the working of the algorithm and the ending of the loop. The following illustrative example will be used:

<table>
<thead>
<tr>
<th>Order Length</th>
<th>Quantity Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5-10 Illustrative example of exact algorithm

In the Table 5-11 below the values of all counters and arrays are displayed step-by-step.

The first statement (page 118) is:

```csharp
If Array_Order_Length(Array_Teller(n)) <= r And Array_Pattern_Small(Array_Teller(n)) < Array_Order_Demand(Array_Teller(n))
```

The second statement (page 119) is:

```csharp
If Array_Teller(n) < Order_Lenghts
```

As seen in the previous section, when the first statement is true, the array `Array_Teller`, the array `Array_Pattern_Big`, the array `Array_Pattern_Small` and the entry space ‘n’ are adapted.
When the first statement is false, the second statement is run. When the second statement is true, the array *Array_Teller* is adapted. When the second statement is false, a new pattern is added to the master problem.
Table 5-11  Values of all counters and arrays displayed step-by-step

<table>
<thead>
<tr>
<th>1st if-statement</th>
<th>Result</th>
<th>Array_Teller</th>
<th>Array_Pattern_Big</th>
<th>Array_Pattern_Small</th>
<th>2nd if-statement</th>
<th>Result</th>
<th>Array_Teller</th>
<th>Master Problem (MP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>60 ≤ 100 and 0 &lt; 1</td>
<td>YES</td>
<td>1</td>
<td>[1]</td>
<td>[ ]</td>
<td>[0 0 0]</td>
<td>60 ≤ 40 and 1 &lt; 1</td>
<td>NO</td>
<td>1 &lt; 3</td>
</tr>
<tr>
<td>20 ≤ 40 and 0 &lt; 2</td>
<td>YES</td>
<td>2</td>
<td>[1 1]</td>
<td>[60]</td>
<td>[1 0 0]</td>
<td>20 ≤ 20 and 1 &lt; 2</td>
<td>YES</td>
<td>1 &lt; 3</td>
</tr>
<tr>
<td>20 ≤ 0 and 2 &lt; 2</td>
<td>NO</td>
<td>3</td>
<td>[1 2 2]</td>
<td>[60 20]</td>
<td>[1 1 0]</td>
<td>20 ≤ 0 and 2 &lt; 2</td>
<td>NO</td>
<td>3 &lt; 3</td>
</tr>
<tr>
<td>15 ≤ 0 and 0 &lt; 2</td>
<td>NO</td>
<td>4</td>
<td>[1 2 2 2]</td>
<td>[60 20 20]</td>
<td>[1 2 0]</td>
<td>15 ≤ 20 and 0 &lt; 2</td>
<td>YES</td>
<td>2 &lt; 3</td>
</tr>
<tr>
<td>15 ≤ 5 and 1 &lt; 2</td>
<td>NO</td>
<td>5</td>
<td>[1 2 3]</td>
<td>[60 20]</td>
<td>[1 1 0]</td>
<td>15 ≤ 20 and 0 &lt; 2</td>
<td>YES</td>
<td>3 &lt; 3</td>
</tr>
<tr>
<td>15 ≤ 40 and 0 &lt; 2</td>
<td>YES</td>
<td>6</td>
<td>[1 2 3 3]</td>
<td>[60 20 15]</td>
<td>[1 1 1]</td>
<td>15 ≤ 25 and 1 &lt; 2</td>
<td>YES</td>
<td>3 &lt; 3</td>
</tr>
<tr>
<td>15 ≤ 10 and 2 &lt; 2</td>
<td>NO</td>
<td>7</td>
<td>[1 2 3 3 3]</td>
<td>[60 20 15 15]</td>
<td>[1 1 2]</td>
<td>15 ≤ 10 and 2 &lt; 2</td>
<td>NO</td>
<td>3 &lt; 3</td>
</tr>
<tr>
<td>20 ≤ 100 and 0 &lt; 2</td>
<td>YES</td>
<td>8</td>
<td>[2]</td>
<td>[ ]</td>
<td>[0 0 0]</td>
<td>20 ≤ 80 and 1 &lt; 2</td>
<td>YES</td>
<td>2 &lt; 3</td>
</tr>
<tr>
<td>20 ≤ 80 and 1 &lt; 2</td>
<td>YES</td>
<td>9</td>
<td>[2 2]</td>
<td>[20]</td>
<td>[0 1 0]</td>
<td>20 ≤ 80 and 1 &lt; 2</td>
<td>YES</td>
<td>2 &lt; 3</td>
</tr>
<tr>
<td>15 ≤ 60 and 0 &lt; 2</td>
<td>YES</td>
<td>10</td>
<td>[2 2 3 3]</td>
<td>[20 20 15]</td>
<td>[0 2 1]</td>
<td>15 ≤ 45 and 1 &lt; 2</td>
<td>YES</td>
<td>3 &lt; 3</td>
</tr>
<tr>
<td>15 ≤ 30 and 2 &lt; 2</td>
<td>NO</td>
<td>11</td>
<td>[2 2 3 3 3]</td>
<td>[20 20 15 15]</td>
<td>[0 2 2]</td>
<td>15 ≤ 30 and 2 &lt; 2</td>
<td>NO</td>
<td>3 &lt; 3</td>
</tr>
<tr>
<td>15 ≤ 80 and 0 &lt; 2</td>
<td>YES</td>
<td>12</td>
<td>[2 3]</td>
<td>[20]</td>
<td>[0 1 0]</td>
<td>15 ≤ 65 and 1 &lt; 2</td>
<td>YES</td>
<td>3 &lt; 3</td>
</tr>
<tr>
<td>15 ≤ 50 and 2 &lt; 2</td>
<td>NO</td>
<td>13</td>
<td>[2 3 3]</td>
<td>[20 15]</td>
<td>[0 1 1]</td>
<td>15 ≤ 50 and 2 &lt; 2</td>
<td>NO</td>
<td>3 &lt; 3</td>
</tr>
<tr>
<td>15 ≤ 100 and 0 &lt; 2</td>
<td>YES</td>
<td>14</td>
<td>[3]</td>
<td>[ ]</td>
<td>[0 0 0]</td>
<td>15 ≤ 85 and 1 &lt; 2</td>
<td>YES</td>
<td>3 &lt; 3</td>
</tr>
<tr>
<td>15 ≤ 75 and 2 &lt; 2</td>
<td>NO</td>
<td>15</td>
<td>[3 3]</td>
<td>[15]</td>
<td>[0 0 1]</td>
<td>15 ≤ 75 and 2 &lt; 2</td>
<td>NO</td>
<td>3 &lt; 3</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.4 Comparing results with use of real data

In this section real data of five arbitrary veranda orders will be used to establish the benefit of the algorithm. In Section 4.4, we have seen that in a first step the roof profile BOM is made. In a second step the long list of roof profiles is divided in different groups according to their colour. In the third step, a further subdivision is made for the different profile types. When these three steps are done, we obtain a table with all packages of same profile and colour, which are ready to be optimized.

The following veranda orders are used: 060223, 060252, 060269, 060287, and 060326. The orders were not chosen completely arbitrary because they all have the same colour. This was done to make the packages of same colour and profile bigger, so that the benefit of the program can better be observed.

The complete table found after executing above three steps can be consulted in Appendix F. The table also contains profiles without colour:

- PVC profiles, which of course are not lacquered;
- brut aluminium profiles, which are not lacquered because they are not visible.

The brut profiles are used for reinforcement of the veranda structure and are placed inside, on the edge of two profiles.

Every package of same colour and profile is a Cutting Stock Problem. All these little CSPs are solved with the exact algorithm, as there are only five different orders, so the packages of same colour and profile are not too big. The results are displayed in Table 5.12 and compared with the situation without the optimizing program.
<table>
<thead>
<tr>
<th>Ref</th>
<th>Profiles needed with optimization</th>
<th>Waste</th>
<th>Profiles needed without optimization</th>
<th>Waste</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKIVER/25MM</td>
<td>1</td>
<td>21.7%</td>
<td>2</td>
<td>60.9%</td>
</tr>
<tr>
<td>B100X50X2</td>
<td>2</td>
<td>18.8%</td>
<td>2</td>
<td>18.8%</td>
</tr>
<tr>
<td>CLA5309 PONS</td>
<td>10</td>
<td>41.5%</td>
<td>10</td>
<td>41.5%</td>
</tr>
<tr>
<td>CLA5512</td>
<td>8</td>
<td>43.1%</td>
<td>8</td>
<td>43.1%</td>
</tr>
<tr>
<td>CLA5512</td>
<td>4</td>
<td>40.4%</td>
<td>4</td>
<td>40.4%</td>
</tr>
<tr>
<td>CLIPSROFIEL01 PONS</td>
<td>4</td>
<td>25.2%</td>
<td>4</td>
<td>25.2%</td>
</tr>
<tr>
<td>CLIPSROFIEL02</td>
<td>1</td>
<td>9.4%</td>
<td>1</td>
<td>9.4%</td>
</tr>
<tr>
<td>CLIPSROFIEL02</td>
<td>3</td>
<td>17.5%</td>
<td>3</td>
<td>17.5%</td>
</tr>
<tr>
<td>E01/25</td>
<td>1</td>
<td>42.4%</td>
<td>2</td>
<td>71.2%</td>
</tr>
<tr>
<td>E02/25</td>
<td>2</td>
<td>49.6%</td>
<td>3</td>
<td>66.4%</td>
</tr>
<tr>
<td>L100X25X2</td>
<td>3</td>
<td>10.2%</td>
<td>4</td>
<td>32.7%</td>
</tr>
<tr>
<td>L30X20X2</td>
<td>1</td>
<td>34.9%</td>
<td>1</td>
<td>34.9%</td>
</tr>
<tr>
<td>L50X25X2</td>
<td>1</td>
<td>23.4%</td>
<td>1</td>
<td>23.4%</td>
</tr>
<tr>
<td>P30X2</td>
<td>10</td>
<td>28.1%</td>
<td>10</td>
<td>28.1%</td>
</tr>
<tr>
<td>P50X2</td>
<td>2</td>
<td>27.9%</td>
<td>2</td>
<td>27.9%</td>
</tr>
<tr>
<td>PVCBUSROND80 LGRIJS</td>
<td>4</td>
<td>15.4%</td>
<td>5</td>
<td>32.3%</td>
</tr>
<tr>
<td>PVCPROFIEL01</td>
<td>8</td>
<td>11.0%</td>
<td>11</td>
<td>35.3%</td>
</tr>
<tr>
<td>SOLIN01</td>
<td>1</td>
<td>26.6%</td>
<td>1</td>
<td>26.6%</td>
</tr>
<tr>
<td>TOLE</td>
<td>2</td>
<td>29.5%</td>
<td>2</td>
<td>29.5%</td>
</tr>
<tr>
<td>TRAVERS01</td>
<td>3</td>
<td>7.1%</td>
<td>6</td>
<td>53.6%</td>
</tr>
<tr>
<td>TRAVERS08</td>
<td>1</td>
<td>40.9%</td>
<td>1</td>
<td>40.9%</td>
</tr>
<tr>
<td>U15X30X15X2</td>
<td>1</td>
<td>33.6%</td>
<td>1</td>
<td>33.6%</td>
</tr>
<tr>
<td>U18X45X20</td>
<td>1</td>
<td>59.7%</td>
<td>1</td>
<td>59.7%</td>
</tr>
<tr>
<td>VER01/25A</td>
<td>7</td>
<td>6.5%</td>
<td>8</td>
<td>18.2%</td>
</tr>
<tr>
<td>VER01/35</td>
<td>12</td>
<td>18.3%</td>
<td>14</td>
<td>30.0%</td>
</tr>
<tr>
<td>VER05/25C</td>
<td>2</td>
<td>25.2%</td>
<td>3</td>
<td>50.1%</td>
</tr>
<tr>
<td>VER05/25E</td>
<td>2</td>
<td>46.2%</td>
<td>2</td>
<td>46.2%</td>
</tr>
<tr>
<td>VER05/25I</td>
<td>2</td>
<td>46.2%</td>
<td>2</td>
<td>46.2%</td>
</tr>
<tr>
<td>VER06</td>
<td>3</td>
<td>9.0%</td>
<td>6</td>
<td>54.5%</td>
</tr>
<tr>
<td>VER15/25</td>
<td>4</td>
<td>8.0%</td>
<td>8</td>
<td>54.0%</td>
</tr>
<tr>
<td>VER34</td>
<td>5</td>
<td>4.5%</td>
<td>6</td>
<td>20.4%</td>
</tr>
<tr>
<td>VER59</td>
<td>3</td>
<td>12.8%</td>
<td>4</td>
<td>34.6%</td>
</tr>
<tr>
<td>VER66/14</td>
<td>8</td>
<td>4.8%</td>
<td>10</td>
<td>23.8%</td>
</tr>
<tr>
<td>VER71 (7000MM)</td>
<td>5</td>
<td>16.0%</td>
<td>6</td>
<td>30.0%</td>
</tr>
<tr>
<td>VER73</td>
<td>2</td>
<td>30.0%</td>
<td>2</td>
<td>30.0%</td>
</tr>
<tr>
<td>VER97</td>
<td>1</td>
<td>60.9%</td>
<td>1</td>
<td>60.9%</td>
</tr>
</tbody>
</table>

Table 5-12 Comparison results with optimizing program and without
Interpretation of the results:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total stock profiles used with the optimizing program</td>
<td>130</td>
</tr>
<tr>
<td>Total stock profiles used without the optimizing program</td>
<td>157</td>
</tr>
<tr>
<td>Reduction of stock profiles used</td>
<td>-17,2%</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total length of pieces cut</td>
<td>672,227 m</td>
</tr>
<tr>
<td>Total length of profiles used with the optimizing program</td>
<td>870,700 m</td>
</tr>
<tr>
<td>Waste with optimizing program</td>
<td>22,8%</td>
</tr>
<tr>
<td>Total length of profiles used without the optimizing program</td>
<td>1053,900 m</td>
</tr>
<tr>
<td>Waste without optimizing program</td>
<td>36,2%</td>
</tr>
<tr>
<td>Waste reduction</td>
<td>-13,4%</td>
</tr>
</tbody>
</table>

The above results have to be read with caution. The fact that profiles can be placed back in the stock after sawing was not taken into account. Of some profiles, only a little piece is needed, which results in a huge waste. For example, profile VER 97 is 6950 mm long, but only order 060287 needs a part of it, which is 2718 mm long, which result in a waste of 60,9 %. Nowadays these profiles are laid back in the stock for further use, if the colour is not a special colour. This is also one of the reasons that special colours, which occur few times, are a lot more expensive; they result in a bigger waste.

If we take into account that profiles where the waste is more than 40 % are laid back in the stock, we get the following results:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total stock profiles used and not laid back with the optimizing program</td>
<td>98</td>
</tr>
<tr>
<td>Total stock profiles used and not laid back without the optimizing program</td>
<td>101</td>
</tr>
<tr>
<td>Reduction of stock profiles used</td>
<td>-3,0%</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total stock profiles laid back in the stock with the optimizing program</td>
<td>32</td>
</tr>
<tr>
<td>Total stock profiles laid back in the stock without the optimizing program</td>
<td>56</td>
</tr>
<tr>
<td>Reduction of profiles laid back in the stock</td>
<td>-42,9%</td>
</tr>
</tbody>
</table>
With the possibility of laying the profiles back in the stock we get a waste reduction of 10,2%. This is less than in the previous case because the situation without the optimizing program gets more advantage of laying the profiles back, because there are more profiles laid back in the stock. The waste of the profiles laid back in the stock is not taken into account in the calculation of the waste, because the waste of these profiles will be used again for further orders. The waste without the optimizing program is now 25,8% which is also the current waste in the company and the waste with the optimizing program is 15,6%.

Another advantage of the program is that fewer profiles have to be laid back in the stock. A reduction of 42,9% is obtained. This results in the possibility of having a bigger stock surface for the profiles that have not been cut yet, and a smaller stock surface for the profiles which are laid back.
Chapter 6

Design Problem

In this chapter we will discuss a design problem, faced by the company in the pre-montage step. In the case of a rising gutter, a combination of a vertical and horizontal cutting angle has to be sawed. A model with 100 x 100 beams is constructed to clarify the 3D visualisation. Subsequently, the vertical cutting angle, the torsion angle and the horizontal cutting angle are calculated in function of the given vertical and horizontal angle of the rising gutter. At the end of the chapter a little example is given to demonstrate the formulas.

6.1 Introduction

Most of the verandas, all the not rectangular ones are pre-assembled after the preparation step (See Chapter 3). The roof is necessarily pre-assembled, because some little profile cuts can not be calculated. The pre-assembled roof is used to take the measurements for the roof material. When the measurements are taken, the aluminium roof is disassembled and sent to the packing step together with the cut roof material.

Some verandas, particularly the verandas with rising gutters, are pre-assembled at real height. In normal pre-assemble cases, the roof is placed on the little wooden blocks and only the roof is pre-assembled. In specific cases with rising gutters, also the window frames and posts are pre-assembled to support the roof, which is placed on its real height. This is necessary because until now the vertical and horizontal cutting edges of the gutter can not be calculated.
Chapter 6  Design Problem

Figure 6-1 An isometric view of the rising gutter

Figure 6-2 A top view of the rising gutter

Figure 6-3 A side view of the rising gutter
Above 3D drawings, made with SolidWorks, give a good view of the flat gutter (blue) and rising gutter (red).

- The first 3D drawing (Figure 6-1) is an isometric view, which gives a general overview.
- The second 3D drawing (Figure 6-2) is a top view, where we get a good view of the horizontal angle, which is chosen 45°.
- The third 3D drawing (Figure 6-3) is a front view and gives a good view of the vertical angle, which is chosen 20°.

The vertical and horizontal angles are given by the client; from these data the construction of the gutter must be started. To make a given vertical angle of 20° for example, two opposite cuts with a vertical angle of 10° are made in the flat and rising gutter respectively. When placing the two gutters against each other, we get a vertical angle of 20°. The horizontal angle of 45° is made with the same method, making two opposite cuts with a horizontal angle of 22°30’. Placing the two gutters against each other gives a horizontal angle of 45°.

The problem is that the cuts are not independent of each other. The above method works fine when the gutter goes only up, i.e. there is only a vertical angle, or when the gutter goes only sideward, i.e. there is only a horizontal angle. In this particular case we have a combination of a vertical and horizontal angle and the above method does not work!

Nowadays, the method is still used for rising gutters. But as it is not as perfect as necessary the cuts made in the sawing step are made a little bit smaller than 10° and 22°30’, for instance 9° and 20° in our example. Because the cuts are smaller, they can be corrected afterwards. This correction finds place in the pre-montage step. The flat and rising gutter are placed on the window and door frames and supported by posts. Subsequently, the desired angle is drawn on the profile with a pencil and with a little sawing machine the angles are increased a little bit. Thereupon the gutters are again placed on the window and door frames. Normally, the angle is still not completely precise. We have to keep in mind that the profiles are long and that a minimum fault on one of the angles gives a fault in the range of centimetres on the other end of the profile. As long as the angles are not accurate enough, they are increased a little bit. This whole process of trial and error can take a while.
The objective of this chapter is to calculate the angles of the vertical and horizontal cuts, so that the gutters can be cut precisely with the biggest sawing machine and no time is lost in the pre-montage area.

6.2 Model with 100 x 100 beams

To calculate the desired angles a model was made with two 100 mm x 100 mm beams. The problem is completely the same, only that we now have flat planes to work with and that the projections will be easier to imagine. The cuts that have to be made to get the desired result are equal if we work with the real gutter or with a simplified gutter consisting of beams.

The letters and triangles on the picture will be used later to calculate the cutting angles.
6.3 Abbreviations

The data given by the client are:

\[ \varepsilon = \text{vertical angle of rising part regard to ground plane} \]
\[ \zeta = \text{horizontal angle of rising part regard to prolongation of flat part} \]

![Figure 6-5 Horizontal angle of rising part](image)

![Figure 6-6 Vertical angle of rising part](image)

The data needed to cut the gutter are:

\[ \beta = \text{vertical cutting angle} \]
\[ \delta = \text{horizontal cutting angle} \]

![Figure 6-7 Vertical horizontal cutting angle](image)

\[ \tau = \text{torsion angle} \]
\[ Z = \text{height and width of the beam (if it is squared)} \]
6.4 Calculation of vertical cutting angle ($\beta$) in function of $\varepsilon$ and $\delta$.

We will first calculate the vertical angle ($\beta$) in function of $\varepsilon$ and $\delta$. $\varepsilon$ is known and $\delta$ will be calculated later. We consider the following triangle on the upper surface of the rising part. This triangle can also be seen on the above pictures of the model.

\[
E = \frac{Z}{\tan(\frac{\pi}{2} - \delta)} = Z \tan(\delta) \quad (6.1)
\]

Subsequently, we consider the little right triangle in the lower part of this big triangle. This smaller triangle is obviously also on the upper surface of the rising part. (Figure 6-4)
We now project D on the extension of the upper surface of the flat part, a surface parallel with the ground surface. This projection gives the following triangle.

When making two opposite cuts with a vertical angle $\beta$, we get an angle $2\beta$ on the lower left corner of above triangle. The height can be worked out as follows:

\[
H = D \sin(2\beta) = E \sin(2\beta) \sin(\pi / 2 - \delta) = Z \sin(2\beta) \sin(\delta)
\]  

(6.3)

We repeat the same projection for E and get the following triangle:

The projection of E gives $\varepsilon$ in the lower left corner. The last two triangles show clearly why the method of making two opposite cuts of half of the desired angle does not work when we have a combination of a vertical and horizontal angle. The desired angle ($\varepsilon$) is twice the cut angle ($\beta$), but the angle made by the projection of the line, perpendicular on the cutting line of the rising part and flat part.
The height is already known, so we can determine $\varepsilon$, using equation’s 6.1 and 6.3:

$$\varepsilon = \arcsin \left( \frac{H}{E} \right) = \arcsin \left[ \sin(2\beta) \cos(\delta) \right] \quad (6.4)$$

The above equitation can be transformed into:

$$\beta = \frac{1}{2} \arcsin \left( \frac{\sin(\varepsilon)}{\cos(\delta)} \right) \quad (6.5)$$

The top view of the below model simplifies imagination of the projections of the edges D and E on the upper surface of the flat part.

Figure 6-8  Top view of the model with 100 x 100 beams
6.5 Calculation of torsion angle ($\tau$) in function of $\varepsilon$ and $\delta$.

In this section the torsion angle $\tau$ of the rising part is computed. The torsion is the consequence of the combination of a vertical and horizontal angle. Figure 6-9 gives a good view of this torsion.

![Figure 6-9 View of the torsion angle of the rising gutter](image)

We consider the following triangle made by the projection of the width of the rising part on a surface parallel with the ground surface. The edges $Z$ and $H$ are the same as in previous triangles.
We now can determine the torsion angle $\tau$:

$$\tau = \arcsin\left[ \sin(\varepsilon) \tan(\delta) \right]$$  \hspace{1cm} (6.6)

### 6.6 Calculation of horizontal cutting edge ($\delta$) in function of $\zeta$ and $\varepsilon$.

$\beta$ was already calculated in function of $\varepsilon$ and $\delta$. Now $\delta$ will be computed in function of $\varepsilon$ and $\zeta$. We consider again the little triangle on the upper surface of the rising part.

![Diagram of triangle DDE']

Projecting this whole triangle on the extension of the upper surface of the flat part gives the following triangle.

![Diagram of triangle D'E']

Out of previous triangles with $D'$ and $E'$ we get:

$$D' = D \cos(2\beta)$$  \hspace{1cm} (6.7)

$$E' = E \cos(\varepsilon)$$

We now can compute $\phi$ in function of $\delta$, $\beta$ and $\varepsilon$, utilizing equation 6.2:
\[
\varphi = \arcsin \left( \frac{D'}{E'} \right) = \arcsin \left( \cos(\delta) \cos(2\beta) \frac{\cos(\varepsilon)}{\cos(\varepsilon)} \right) \quad (6.8)
\]

When observing the top view (Figure 6.8) we can write:
\[
\pi = \zeta + \varphi + \left( \frac{\pi}{2} - \delta \right) \quad (6.9)
\]

This equitation can be transformed to the following equitation using equation 6.8:
\[
\delta = \zeta + \arcsin \left( \frac{\cos(\delta) \cos(2\beta)}{\cos(\varepsilon)} \right) - \frac{\pi}{2} \quad (6.10)
\]

A transcendental equitation is obtained, \( \delta \) is function of \( \delta \). Above equitation cannot be solved analytically to \( \delta \). Because the function \( \arcsin \) can be written as a convergent series, the whole right hand side can be written as a convergent series. This signifies that \( \delta \) can be found after some iterations.

\[
\delta - \zeta + \frac{\pi}{2} = \arcsin \left[ \cos(\delta) \cos \left( \arcsin \left( \frac{\sin(\varepsilon)}{\cos(\delta)} \right) \right) \right] \quad (6.11)
\]

In the above equation \( \zeta \) and \( \varepsilon \) are known. After some iterations, we will find \( \delta \) such that the right hand side equals \( \zeta \).
6.7 An example

In this section we will elaborate a little example. The client wants a gutter with:

- a vertical angle ($\varepsilon$) of 20°,
- a horizontal angle ($\zeta$) of 45°.

Putting the data in above formulas gives:

- a vertical cutting angle ($\beta$) of 10.80°
- a horizontal cutting angle ($\delta$) of 21.76°
- a torsion angle ($\tau$) of 7.85°
Conclusions

We studied two main problems the company is facing: a waste reduction problem and a design problem.

For the waste reduction problem we suggest the following procedure: making the roof profile BOM; dividing the long list of roof profiles in different groups according to their colour; making a further subdivision for the different profile types; and sawing the packages of the same profile and colour in an optimized way.

Each package of the same profile and colour is a cutting stock problem, ready to be optimized. We propose two algorithms: a low demand column generation algorithm and an exact algorithm.

The low demand column generation algorithm is an improvement of Gilmore and Gomory’s algorithm for low demands. The knapsack problem was adapted; making sure that in every new pattern the number of cut pieces of a particular length is smaller or equal than the total number of pieces needed. An upper bound was placed on each variable. The initial set of patterns was also changed. The algorithm is as follows:

1. Generate an initial set of patterns for the master problem.
2. Solve the master problem with the simplex algorithm.
3. Solve the knapsack problem by using the shadow prices ($\pi_i$) of the master problem.
4. If the objective value of the knapsack problem is negative, go to step 2 and add the new pattern to the master problem.
5. Solve the master problem with all generated patterns and without relaxation.

Because of the delayed column generation, it is still a heuristic.

Exact algorithms in the literature change the way in which the Branch-and-Bound is done. Our exact algorithm gives an exact solution without changing the Branch-and-Bound code.
For the specific case of the company, demands are the number of profiles ordered with the same length. As the lengths are measured up to the millimetre and all verandas are different, it is clear that the quantities of the demands will be low. Keeping this in mind, the patterns can first be generated. The patterns are generated on a smart way, not all the patterns are generated but only a little portion of it, used to solve the cutting stock problem exactly. Only ‘full’ and ‘required’ patterns are generated. A ‘full’ pattern is a pattern in which no more pieces can be added. A ‘required’ pattern is a pattern where cut pieces do not exceed quantity ordered. While generating only ‘full’ and ‘required’ patterns a pattern reduction of more than 75% was reached.

The low demand column generation algorithm will be more suitable for the future, when the company will be able to combine more veranda orders. Nowadays, because of space limitation and in default of an automatic sawing machine with labeller, only three veranda orders can be combined. The exact algorithm will solve these little problems fast enough and the optimal solution is ensured.

We also inquired the company about the possibility of combining two pieces with an angle, placing two same profiles on each other, placing two different profiles on each other. This last combination is possible for some profiles that can be telescoped into each other. The other combinations are not feasible because the profiles are not symmetric and can not be turned on the table.

The two algorithms were programmed in Visual Basic, and the CSP was solved utilizing lp_solve. To establish the benefit of the program, real company data was used to combine five veranda orders. The results were compared with the situation without the optimizing program. The waste without the optimizing program is 25.8% which is also the current waste in the company and the waste with the optimizing program is 15.6%. We obtain a waste reduction of 10.2%. Without the possibility of laying back profiles in the stock for further use, we even get a waste reduction of 17.2%. Another advantage of the program is that fewer profiles have to be laid back in the stock. A reduction of 42.9% is obtained, which results in a better exploit of the warehouse surface.

The second problem we investigated was a design problem occurring when cutting the angles of a rising gutter. The vertical cutting angle ($\beta$), the torsion angle ($\tau$) and the horizontal
cutting angle ($\delta$) were calculated in function of the given vertical angle ($\varepsilon$) and horizontal angle ($\zeta$) of the rising gutter. The calculation results in a profitable gain of production time for the company.
References


Appendix A

List of all profiles

SERIE VER

VER 01.25

VER 01.25

VER 02.25

VER 03

VER 04

VER 06

VER 02

VER 04

VER 06

VER 08

VER 05.25C

VER 05.25C

VER 05.24

VER 05.24

VER 06

VER 06

VER 06

VER 06

VER 20/A

VER 20/A

VER 34

VER 34

VER 58

VER 58
Appendix A

List of all profiles
Appendix A  List of all profiles

SERIE TO

TO 01  TO 11  TO 20  TO 24  TO 33 A
TO 31  TO 91  TO 92  TO 94  TO 4138

SERIE CP

CP 01  CP 02  CP 03  CP 04  CP 05  CP 06
CP 07  CP 08  CP 10  CP 11  CP 12

SERIE D

D 1/B  D 2  D 4/B

DIVERS

SOLIN 01  SP-04  SP-02
Appendix B

Detailed plans of the roof profile types
Appendix B
Detailed plans of the roof profile types
Appendix B

Detailed plans of the roof profile types
Appendix C

VB-code low demand column generation algorithm

This Appendix gives the complete Visual Basic code for the exact algorithm developed in Section 5.2. The code is easier to read because of the landscape page and the absent explanation text. In the end the extra Functions and Subs are given.

Option Explicit
Public lpsolve As lpsolve55
Public Sub Cutting_Stock_Problem()
Cleanup
Set lpsolve = New lpsolve55
Debug.Print CurDir$
lpsolve.Init Application.ActiveWorkbook.Path

Dim i As Long
Dim j As Long
Dim k As Long

'************************** Profile Lengths **************************
Dim Profile_Length As Integer
Profile_Length = Application.Sheets("Blad1").Cells(2, "C")

Dim Order_Lengths As Integer
Dim Array_Order_Length() As Double

Order_Lengths = Application.Sheets("Blad1").Cells(4, "B")
ReDim Array_Order_Length(Order_Lengths)
For i = 1 To Order_Lengths
    Array_Order_Length(i) = Application.Sheets("Blad1").Cells(6 + i - 1, "B")
Next

'****************** Demands **************************************
Dim Order_Demands As Integer
Dim Array_Order_Demand() As Integer
Order_Demands = Application.Sheets("Blad1").Cells(4, "B")
ReDim Array_Order_Demand(Order_Demands)
For i = 1 To Order_Demands
    Array_Order_Demand(i) = Application.Sheets("Blad1").Cells(6 + i - 1, "C")
Next

'****************** Define Master_Problem *************************
Dim Array_Row_Zero() As Double
ReDim Array_Row_Zero(Order_Lengths)
For i = 1 To Order_Lengths
    Array_Row_Zero(i) = 1
Next

Dim Master_Problem As Long
With lpsolve
    Master_Problem = .make_lp(0, Order_Lengths)
    .set_outputfile Master_Problem, Application.ActiveWorkbook.Path & "\Master_Problem.txt"
    .set_add_rowmode Master_Problem, True
    .set_obj_fn Master_Problem, Array_Row_Zero(0)
    Dim ColNo As Long
    Dim SparseRow As Double
    ColNo = 1
    For i = 1 To Order_Lengths
        SparseRow = Min(Int(Profile_Length / Array_Order_Length(i)), Array_Order_Demand(i))
        .add_constraintex Master_Problem, 1, SparseRow, ColNo, GE, Array_Order_Demand(i)
        ColNo = ColNo + 1
    Next
    .set_add_rowmode Master_Problem, False
End With

'****************** Define Knapsack ******************************
Dim Knapsack As Long
With lpsolve
    Knapsack = .make_lp(0, Order_Lengths)
    .set_outputfile Knapsack, Application.ActiveWorkbook.Path & "\Knapsack.txt"
    .set_maxim Knapsack
    .set_add_rowmode Knapsack, True
    .add_constraint Knapsack, Array_Order_Length(0), LE, Profile_Length
    For i = 1 To Order_Lengths
        .set_upbo Knapsack, i, Array_Order_Demand(i)
    Next
    .set_add_rowmode Knapsack, False
    For i = 1 To Order_Lengths
        .set_int Knapsack, i, True
    Next
End With

'********************* Cutting Stock Algorithm *********************
Dim Obj As Double
Dim Flag As Boolean

Dim Array_New_Column() As Double
ReDim Array_New_Column(Order_Lengths)

Dim Array_Duals() As Double

With lpsolve
    Obj = 10
    Flag = False
    While Obj > 1.00001
        If Flag = True Then
            ReDim Array_New_Column(1 To .get_Ncolumns(Knapsack) + 1)
            .get_variables Knapsack, Array_New_Column(1)
            For i = Order_Lengths + 1 To 2 Step -1
                Array_New_Column(i) = Array_New_Column(i - 1)
            Next
        End If
        Flag = True
    End While
End With
Appendix C

VB-code for low demand column generation

Next
Array_New_Column(1) = 1
    .add_column Master_Problem, Array_New_Column(1)
End If
    .solve Master_Problem
ReDim Array_Duals(0 To .get_Ncolumns(Master_Problem) + .get_Nrows(Master_Problem))
    .get_dual_solution Master_Problem, Array_Duals(0)
    .set_add_rowmode Knapsack, True
        .set_obj_fn Knapsack, Array_Duals(0)
    .set_add_rowmode Knapsack, False
    .solve Knapsack
Obj = .get_objective(Knapsack)
Flag = True
End With

'***************************************** Solve Master Problem integer***************
With lp.solve
    For i = 0 To .get_Ncolumns(Master_Problem)
        .set_int Master_Problem, i, True
    Next
    .solve Master_Problem
End With

'***************************************** Write solution to Excel ***************
Dim Array_Cut() As Double
Dim Array_Patterns() As Double
Dim Array_Quantity_Cut() As Double
Dim ProfileNo As Integer
Dim ProfileNoBis As Integer
Dim Used_Length As Integer
Dim Waste As Integer
Dim Percentage_Waste As Double

With lpsolve
    ReDim Array_Cut(1 To .get_Ncolumns(Master_Problem))
    .get_variables Master_Problem, Array_Cut(1)

    ReDim Array_Patterns(.get_Nrows(Master_Problem), .get_objective(Master_Problem))

    For j = 1 To .get_Ncolumns(Master_Problem)
        If Array_Cut(j) <> 0 Then
            For k = Array_Cut(j) To 1 Step -1
                ProfileNo = ProfileNo + 1
                .get_column Master_Problem, j, Array_Patterns(0, ProfileNo)
            Next
        End If
    Next

    ReDim Array_Quantity_Cut(Order_Lengths)

    For i = 1 To Order_Lengths
        For j = 1 To ProfileNo
            Array_Quantity_Cut(i) = Array_Quantity_Cut(i) + Array_Patterns(i, j)
        Next
    Next

    For i = 1 To Order_Lengths
        If Array_Quantity_Cut(i) > Array_Order_Demand(i) Then
            For k = Array_Quantity_Cut(i) To Array_Order_Demand(i) + 1 Step -1
                Flag = False
                For j = 1 To ProfileNo
                    If Array_Patterns(i, j) > 0.01 And Flag = False Then
                        Array_Patterns(i, j) = Array_Patterns(i, j) - 1
                        Flag = True
                    End If
                Next
            Next
        End If
    Next
End If
Next

For j = 1 To ProfileNo
    Flag = False
    For i = 1 To Order_Lengths
        If Array_Patterns(i, j) <> Array_Patterns(i, j - 1) And Flag = False Then
            Flag = True
            ProfileNoBis = ProfileNoBis + 1
            k = 0
        End If
    Next

    If Flag = True Then
        Application.Sheets("Blad1").Cells(4, Chr(65 + 5 + ProfileNoBis)) = 1
        Application.Sheets("Blad1").Cells(5, Chr(65 + 5 + ProfileNoBis)) = " Pattern " & ProfileNoBis
        Used_Length = 0
        For i = 1 To Order_Lengths
            Application.Sheets("Blad1").Cells(5 + i, Chr(65 + 5 + ProfileNoBis)) = Array_Patterns(i, j)
            Used_Length = Used_Length + (Array_Patterns(i, j) * Array_Order_Length(i))
        Next
        Waste = Waste + (Profile_Length - Used_Length)
    Else
        Application.Sheets("Blad1").Cells(4, Chr(65 + 5 + ProfileNoBis)) = 2 + k
        k = k + 1
        Waste = Waste + (Profile_Length - Used_Length)
    End If
Next

Percentage_Waste = (Waste / (Profile_Length * .get_objective(Master_Problem))) * 100
Application.Sheets("Blad1").Cells(2, "G") = .get_objective(Master_Problem)
Application.Sheets("Blad1").Cells(3, "G") = Waste
Application.Sheets("Blad1").Cells(3, "H") = Round(Percentage_Waste, 3) / 100
.print_lp Master_Problem
.print_lp Knapsack
End With

Set lpsolve = Nothing
Appendix C

End Sub

******************************************************************

Private Sub CommandButton1_Click()
    Cutting_Stock_Problem
End Sub

Function Min(a As Integer, b As Integer) As Double
    If b - a > 0 Then
        Min = a
    Else
        Min = b
    End If
End Function

Sub Cleanup()
    Range("G2:G1000").Select
    Range(Selection, Selection.End(xlToRight)).Select
    Selection.ClearContents
    Range("G2").Select
End Sub
Appendix D

Example output file master problem

In Section 5.2.9, the following example was used to demonstrate the Visual Basic Program:

<table>
<thead>
<tr>
<th>Order Length</th>
<th>Quantity Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>2</td>
</tr>
<tr>
<td>60</td>
<td>3</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5-5 gives the solution found by the low demand column generation algorithm.

This appendix shows the output file of the master problem. Each time the master problem is solved the solution is written away in the output file. The last time the master problem is solved with integer variables. First a relaxed solution is found and subsequently a feasible solution. The Branch-and-Bound nodes needed to find the integer solution are also given. In the beginning of the output file the model size is analysed. In the end the complete matrix A is displayed. The first five columns are the initial patterns, the following four columns are generated by solving the knapsack problem. The first row is the objective function. The last column is the right hand sides of the constraints.

Model name: ' ' - run #1
Objective: Minimize(R0)

SUBMITTED
Model size: 5 constraints, 5 variables, 5 non-zeros.
Sets: 0 GUB, 0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Optimal solution
Excellent numeric accuracy ||*|| = 0

MEMO: lp solve version 5.5.0.5 for 32 bit OS, with 64 bit REAL variables. In the total iteration count 5, 0 (0.0%) were bound flips.
There were 0 refactorizations, 0 triggered by time and 0 by density.
... on average 5.0 major pivots per refactorization.
The largest [LUSOL v2.2.1.0] fact(B) had 6 NZ entries, 1.0x largest basis.
The constraint matrix inf-norm is 3, with a dynamic range of 3.
Time to load data was 0.000 seconds, presolve used 0.062 seconds,
... 0.016 seconds in simplex solver, in total 0.078 seconds.
Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

159
Optimal solution 7.33333333333 after 1 iter.
Excellent numeric accuracy ||*|| = 0

MEMO: lp_solve version 5.5.0.5 for 32 bit OS, with 64 bit REAL variables.
In the total iteration count 1, 0 (0.0%) were bound flips.
There were 1 refactorizations, 0 triggered by time and 0 by density.
... on average 1.0 major pivots per refactorization.
The largest [LUSOL v2.2.1.0] fact(B) had 11 NZ entries, 1.0x largest basis.
The constraint matrix inf-norm is 3, with a dynamic range of 3.
Time to load data was 0.078 seconds, presolve used 0.000 seconds,
0.000 seconds in simplex solver, in total 0.078 seconds.
Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Optimal solution 7 after 1 iter.
Excellent numeric accuracy ||*|| = 0

MEMO: lp_solve version 5.5.0.5 for 32 bit OS, with 64 bit REAL variables.
In the total iteration count 1, 0 (0.0%) were bound flips.
There were 1 refactorizations, 0 triggered by time and 0 by density.
... on average 1.0 major pivots per refactorization.
The largest [LUSOL v2.2.1.0] fact(B) had 12 NZ entries, 1.0x largest basis.
The constraint matrix inf-norm is 3, with a dynamic range of 3.
Time to load data was 0.078 seconds, presolve used 0.000 seconds,
0.000 seconds in simplex solver, in total 0.078 seconds.
Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Optimal solution 6 after 2 iter.
Excellent numeric accuracy ||*|| = 0

MEMO: lp_solve version 5.5.0.5 for 32 bit OS, with 64 bit REAL variables.
In the total iteration count 2, 0 (0.0%) were bound flips.
There were 1 refactorizations, 0 triggered by time and 0 by density.
... on average 2.0 major pivots per refactorization.
The largest [LUSOL v2.2.1.0] fact(B) had 12 NZ entries, 1.0x largest basis.
The constraint matrix inf-norm is 3, with a dynamic range of 3.
Time to load data was 0.078 seconds, presolve used 0.000 seconds,
0.000 seconds in simplex solver, in total 0.078 seconds.
Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Optimal solution 6 after 1 iter.
Excellent numeric accuracy ||*|| = 0

MEMO: lp_solve version 5.5.0.5 for 32 bit OS, with 64 bit REAL variables.
In the total iteration count 1, 0 (0.0%) were bound flips.
There were 1 refactorizations, 0 triggered by time and 0 by density.
... on average 1.0 major pivots per refactorization.
The largest [LUSOL v2.2.1.0] fact(B) had 15 NZ entries, 1.1x largest basis.
The constraint matrix inf-norm is 3, with a dynamic range of 3.
Time to load data was 0.078 seconds, presolve used 0.000 seconds,
0.000 seconds in simplex solver, in total 0.078 seconds.
set int: Column 0 out of range
Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.
**Relaxed solution** 6 after 0 iter is B&B base.

**Feasible solution** 7 after 3 iter, 3 nodes (gap 14.3%)

**Improved solution** 6 after 5 iter, 5 nodes (gap 0.0%)

**Optimal solution** 6 after 5 iter, 5 nodes (gap 0.0%).

*Excellent numeric accuracy ||*|| = 0*

**MEMO:** lp solver version 5.5.0.5 for 32 bit OS, with 64 bit REAL variables.
In the total iteration count 5, 0 (0.0%) were bound flips.
There were 3 refactorizations, 0 triggered by time and 0 by density.
... on average 1.7 major pivots per refactorization.
The largest [LUSOL v2.2.1.0] fact(B) had 14 NZ entries, 1.0x largest basis.
The maximum B&B level was 4, 0.2x MIP order, 3 at the optimal solution.
The constraint matrix inf-norm is 3, with a dynamic range of 3.
Time to load data was 0.078 seconds, presolve used 0.000 seconds,
... 0.000 seconds in simplex solver, in total 0.078 seconds.

**Model name:**

<table>
<thead>
<tr>
<th>Minimize</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>C8</th>
<th>C9</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1  &gt;=</td>
</tr>
<tr>
<td>R2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0  &gt;=</td>
</tr>
<tr>
<td>R3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0  &gt;=</td>
</tr>
<tr>
<td>R4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0  &gt;=</td>
</tr>
<tr>
<td>R5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2  &gt;=</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Int</th>
<th>Int</th>
<th>Int</th>
<th>Int</th>
<th>Int</th>
<th>Int</th>
<th>Int</th>
<th>Int</th>
</tr>
</thead>
<tbody>
<tr>
<td>upbo</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>lowbo</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

161
Appendix E

VB-code exact algorithm

This Appendix gives the complete Visual Basic code for the exact algorithm developed in Section 5.3. The code is easier to read because of the landscape page and the absent explanation text. In the end the extra Functions and Subs are given.

Option Explicit
Public lpsolve As lpsolve55
Public Sub Cutting_Stock_Problem()
  Cleanup
  Sort_Order_Lenghts
  Set lpsolve = New lpsolve55
  Debug.Print CurDir$
  lpsolve.Init Application.ActiveWorkbook.Path
  Dim i As Long
  Dim j As Long
  Dim k As Long
  Dim r As Long
  Dim n As Integer

  '************************ Profile Lengths *********************************
  Dim Profile_Length As Integer
  Profile_Length = Application.Sheets("Blad1").Cells(2, "C")
  Dim Order_Lenghts As Integer
  Dim Array_Order_Length() As Double
Appendix E

VB-code for exact algorithm

Order_Lengths = Application.Sheets("Blad1").Cells(4, "B")
ReDim Array_Order_Length(Order_Lengths)

For i = 1 To Order_Lengths
    Array_Order_Length(i) = Application.Sheets("Blad1").Cells(6 + i - 1, "B")
Next

'****************** Demands **************************************************
Dim Order_Demands As Integer
Dim Array_Order_Demand() As Integer

Order_Demands = Application.Sheets("Blad1").Cells(4, "B")
ReDim Array_Order_Demand(Order_Demands)

For i = 1 To Order_Demands
    Array_Order_Demand(i) = Application.Sheets("Blad1").Cells(6 + i - 1, "C")
Next

'****************** Define Master_Problem *************************************
Dim Array_New_Column() As Double
Dim Array_Teller() As Double
Dim Array_Pattern_Big() As Double
Dim Array_Pattern_Small() As Double
Dim Flag As Boolean
Dim Master_Problem As Long
With lpsolve
    Master_Problem = .make_lp(Order_Lengths, 0)
    .set_outputfile Master_Problem, Application.ActiveWorkbook.Path & "\Master_Problem.txt"
    ReDim Array_Teller(Int(Profile_Length / Array_Order_Length(Order_Lengths)) + 1)
    Array_Teller(1) = 1
    ReDim Array_Pattern_Big(Int(Profile_Length / Array_Order_Length(Order_Lengths)) + 1)
    ReDim Array_Pattern_Small(Order_Lengths)
    n = 1
    While n > 0

163
r = Profile_Length
For i = 1 To n - 1
    r = r - Array_Pattern_Big(i)
Next

If Array_Order_Length(Array_Teller(n)) <= r And Array_Pattern_Small(Array_Teller(n)) < Array_Order_Demand(Array_Teller(n)) Then
    Array_Teller(n + 1) = Array_Teller(n)
    Array_Pattern_Big(n) = Array_Order_Length(Array_Teller(n))
    Array_Pattern_Small(Array_Teller(n)) = Array_Pattern_Small(Array_Teller(n)) + 1
    n = n + 1
Else

    If Array_Teller(n) < Order_Lengths Then
        Array_Teller(n) = Array_Teller(n) + 1
    Else
        ReDim Array_New_Column(1 To Order_Lengths + 1)
        ReDim Array_Pattern_Small(Order_Lengths)
        For i = 1 To n - 1
            Flag = False
            For j = 1 To Order_Lengths
                If Array_Pattern_Big(i) = Array_Order_Length(j) And Flag = False Then
                    Array_New_Column(j + 1) = Array_New_Column(j + 1) + 1
                    Flag = True
                End If
            Next
        Next
        Array_New_Column(1) = 1
        .add_column Master_Problem, Array_New_Column(1)
k = 1
While Array_Teller(n - k) = Order_Lenghts
    k = k + 1
Wend

Array_Teller(n - k) = Array_Teller(n - k) + 1
n = n - k

For i = 1 To n - 1
    Flag = False
    For j = 1 To Order_Lenghts
        If Array_Pattern_Big(i) = Array_Order_Length(j) And Flag = False Then
            Array_Pattern_Small(j) = Array_Pattern_Small(j) + 1
            Flag = True
        End If
    Next
End If
Next

For i = 1 To .get_Ncolumns(Master_Problem)
    .set_int Master_Problem, i, True
Next

For i = 1 To Order_Lenghts
    .set_rh Master_Problem, i, Array_Order_Demand(i)
    .set_constr_type Master_Problem, i, GE
Next
.solver Master_Problem
End With

'************************** Write solution to Excel **************************
Dim Array_Cut() As Double
Dim Array_Patterns() As Double
Dim Array_Quantity_Cut() As Double
Dim ProfileNo As Integer
Dim ProfileNoBis As Integer

Dim Used_Length As Integer
Dim Waste As Integer
Dim Percentage_Waste As Double

With lpsolve
    ReDim Array_Cut(1 To .get_Ncolumns(Master_Problem))
    .get_variables Master_Problem, Array_Cut(1)

    ReDim Array_Patterns(.get_Nrows(Master_Problem), .get_objective(Master_Problem))

    For j = 1 To .get_Ncolumns(Master_Problem)
        If Array_Cut(j) <> 0 Then
            For k = Array_Cut(j) To 1 Step -1
                ProfileNo = ProfileNo + 1
                .get_column Master_Problem, j, Array_Patterns(0, ProfileNo)
            Next
        End If
    Next

    ReDim Array_Quantity_Cut(Order_Lengths)

    For i = 1 To Order_Lengths
        For j = 1 To ProfileNo
            Array_Quantity_Cut(i) = Array_Quantity_Cut(i) + Array_Patterns(i, j)
        Next
    Next

    For i = 1 To Order_Lengths
        If Array_Quantity_Cut(i) > Array_Order_Demand(i) Then
            For k = Array_Quantity_Cut(i) To Array_Order_Demand(i) + 1 Step -1
                Flag = False
                For j = 1 To ProfileNo
                    If Array_Patterns(i, j) > 0.01 And Flag = False Then
                        Array_Patterns(i, j) = Array_Patterns(i, j) - 1
                        Flag = True
                    End If
                Next
            End If
        End If
    Next
Next
    Next
End If
Next

For j = 1 To ProfileNo
    Flag = False
    For i = 1 To Order_Lengths
        If Array_Patterns(i, j) <> Array_Patterns(i, j - 1) And Flag = False Then
            Flag = True
            ProfileNoBis = ProfileNoBis + 1
            k = 0
        End If
    Next
    If Flag = True Then
        Application.Sheets("Blad1").Cells(4, Chr(65 + 5 + ProfileNoBis)) = 1
        Application.Sheets("Blad1").Cells(5, Chr(65 + 5 + ProfileNoBis)) = " Pattern " & ProfileNoBis
        Used_Length = 0
        For i = 1 To Order_Lengths
            Application.Sheets("Blad1").Cells(5 + i, Chr(65 + 5 + ProfileNoBis)) = Array_Patterns(i, j)
            Used_Length = Used_Length + (Array_Patterns(i, j) * Array_Order_Length(i))
        Next
        Waste = Waste + (Profile_Length - Used_Length)
    Else
        Application.Sheets("Blad1").Cells(4, Chr(65 + 5 + ProfileNoBis)) = 2 + k
        k = k + 1
        Waste = Waste + (Profile_Length - Used_Length)
    End If
Next

Percentage_Waste = (Waste / (Profile_Length * .get_objective(Master_Problem))) * 100

Application.Sheets("Blad1").Cells(2, "G") = .get_objective(Master_Problem)
Application.Sheets("Blad1").Cells(3, "G") = Waste
Application.Sheets("Blad1").Cells(3, "H") = Round(Percentage_Waste, 3) / 100
.print_lp Master_Problem
.print_lp Knapsack
End With
Set lpsolve = Nothing
End Sub

*****************************************************************************

Private Sub CommandButton1_Click()
    Cutting_Stock_Problem
End Sub

Function Min(a As Integer, b As Integer) As Double
    If b - a > 0 Then
        Min = a
    Else
        Min = b
    End If
End Function

Sub Cleanup()
    Range("G2:G1000").Select
    Range(Selection, Selection.End(xlToRight)).Select
    Selection.ClearContents
    Range("G2").Select
End Sub

Sub Sort_Order_Lengths()
    Range("B5:C20").Sort Key1:=Range("B5"), Order1:=xlDescending, Header:=xlGuess, OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom, DataOption1:=xlSortNormal
End Sub
Appendix F

Table with packages of same profile and colour

The table below gives all the packages of same profile and colour for the following five veranda orders: 060223, 060252, 060269, 060287, and 060326. Every package is a Cutting Stock Problem and can be optimized. These real data are used to demonstrate the benefit of the optimization program. The results are displayed in Section 5.4.

<table>
<thead>
<tr>
<th>Number of profiles</th>
<th>312</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ref</td>
<td></td>
</tr>
<tr>
<td>Colour</td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td></td>
</tr>
<tr>
<td>File</td>
<td></td>
</tr>
<tr>
<td>Max. Length</td>
<td></td>
</tr>
<tr>
<td>AKIVER/25MM</td>
<td>1015</td>
</tr>
<tr>
<td>AKIVER/25MM</td>
<td>1015</td>
</tr>
<tr>
<td>B100X50X2</td>
<td>1015</td>
</tr>
<tr>
<td>B100X50X2</td>
<td>1015</td>
</tr>
<tr>
<td>B100X50X2</td>
<td>1015</td>
</tr>
<tr>
<td>B100X50X2</td>
<td>1015</td>
</tr>
<tr>
<td>CLA5309 PONS</td>
<td>3952</td>
</tr>
<tr>
<td>CLA5309 PONS</td>
<td>3600</td>
</tr>
<tr>
<td>CLA5309 PONS</td>
<td>3484</td>
</tr>
<tr>
<td>CLA5309 PONS</td>
<td>1200</td>
</tr>
<tr>
<td>CLA5512</td>
<td>1015</td>
</tr>
<tr>
<td>CLA5512</td>
<td>8004M</td>
</tr>
<tr>
<td>CLA5512</td>
<td>8004M</td>
</tr>
<tr>
<td>CLA5512</td>
<td>8004M</td>
</tr>
<tr>
<td>CLIPS PROFI E01 PONS</td>
<td>3150</td>
</tr>
<tr>
<td>CLIPS PROFI E01 PONS</td>
<td>3050</td>
</tr>
<tr>
<td>CLIPS PROFI E01 PONS</td>
<td>2850</td>
</tr>
<tr>
<td>CLIPS PROFI E01 PONS</td>
<td>2700</td>
</tr>
<tr>
<td>CLIPS PROFI E02</td>
<td>8004M</td>
</tr>
<tr>
<td>CLIPS PROFI E02</td>
<td>1015</td>
</tr>
<tr>
<td>CLIPS PROFI E02</td>
<td>1015</td>
</tr>
<tr>
<td>CLIPS PROFI E02</td>
<td>1015</td>
</tr>
<tr>
<td>E01/25</td>
<td>1015</td>
</tr>
<tr>
<td>E01/25</td>
<td>1015</td>
</tr>
<tr>
<td>E02/25</td>
<td>1015</td>
</tr>
<tr>
<td>E02/25</td>
<td>1015</td>
</tr>
<tr>
<td>E02/25</td>
<td>1015</td>
</tr>
<tr>
<td>Description</td>
<td>Code</td>
</tr>
<tr>
<td>----------------------</td>
<td>------</td>
</tr>
<tr>
<td>L100X25X2</td>
<td>1015</td>
</tr>
<tr>
<td>L100X25X2</td>
<td>1015</td>
</tr>
<tr>
<td>L100X25X2</td>
<td>1015</td>
</tr>
<tr>
<td>L100X25X2</td>
<td>1015</td>
</tr>
<tr>
<td>L100X25X2</td>
<td>1015</td>
</tr>
<tr>
<td>L30X20X2</td>
<td>1015</td>
</tr>
<tr>
<td>L50X25X2</td>
<td>1015</td>
</tr>
<tr>
<td>P30X2</td>
<td>1015</td>
</tr>
<tr>
<td>P30X2</td>
<td>1015</td>
</tr>
<tr>
<td>P30X2</td>
<td>1015</td>
</tr>
<tr>
<td>P30X2</td>
<td>1015</td>
</tr>
<tr>
<td>P30X2</td>
<td>1015</td>
</tr>
<tr>
<td>P30X2</td>
<td>1015</td>
</tr>
<tr>
<td>P30X2</td>
<td>1015</td>
</tr>
<tr>
<td>P30X2</td>
<td>1015</td>
</tr>
<tr>
<td>P30X2</td>
<td>1015</td>
</tr>
<tr>
<td>P50X2</td>
<td>1015</td>
</tr>
<tr>
<td>P50X2</td>
<td>1015</td>
</tr>
<tr>
<td>PVCBIUSROND80 LGRIJS</td>
<td>1015</td>
</tr>
<tr>
<td>PVCBIUSROND80 LGRIJS</td>
<td>1015</td>
</tr>
<tr>
<td>PVCBIUSROND80 LGRIJS</td>
<td>1015</td>
</tr>
<tr>
<td>PVCBIUSROND80 LGRIJS</td>
<td>1015</td>
</tr>
<tr>
<td>PVCBIUSROND80 LGRIJS</td>
<td>1015</td>
</tr>
<tr>
<td>PVCPROFIEL01</td>
<td>1015</td>
</tr>
<tr>
<td>PVCPROFIEL01</td>
<td>1015</td>
</tr>
<tr>
<td>PVCPROFIEL01</td>
<td>1015</td>
</tr>
<tr>
<td>PVCPROFIEL01</td>
<td>1015</td>
</tr>
<tr>
<td>PVCPROFIEL01</td>
<td>1015</td>
</tr>
<tr>
<td>PVCPROFIEL01</td>
<td>1015</td>
</tr>
<tr>
<td>PVCPROFIEL01</td>
<td>1015</td>
</tr>
<tr>
<td>PVCPROFIEL01</td>
<td>1015</td>
</tr>
<tr>
<td>PVCPROFIEL01</td>
<td>1015</td>
</tr>
<tr>
<td>SOLIN01</td>
<td>1015</td>
</tr>
<tr>
<td>SOLIN01</td>
<td>1015</td>
</tr>
<tr>
<td>TOLE</td>
<td>1015</td>
</tr>
<tr>
<td>TOLE</td>
<td>1015</td>
</tr>
<tr>
<td>TRAVERS01</td>
<td>1015</td>
</tr>
<tr>
<td>TRAVERS01</td>
<td>1015</td>
</tr>
<tr>
<td>TRAVERS01</td>
<td>1015</td>
</tr>
<tr>
<td>TRAVERS01</td>
<td>1015</td>
</tr>
<tr>
<td>TRAVERS01</td>
<td>1015</td>
</tr>
<tr>
<td>TRAVERS08</td>
<td>1015</td>
</tr>
<tr>
<td>U15X30X15X2</td>
<td>1015</td>
</tr>
<tr>
<td>U18X45X20</td>
<td>1015</td>
</tr>
<tr>
<td>Code</td>
<td>Ver</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td>VER01/25A</td>
<td>1015</td>
</tr>
<tr>
<td>VER01/25A</td>
<td>1015</td>
</tr>
<tr>
<td>VER01/25A</td>
<td>1015</td>
</tr>
<tr>
<td>VER01/25A</td>
<td>1015</td>
</tr>
<tr>
<td>VER01/25A</td>
<td>1015</td>
</tr>
<tr>
<td>VER01/35</td>
<td>1015</td>
</tr>
<tr>
<td>VER01/35</td>
<td>1015</td>
</tr>
<tr>
<td>VER01/35</td>
<td>1015</td>
</tr>
<tr>
<td>VER01/35</td>
<td>1015</td>
</tr>
<tr>
<td>VER01/35</td>
<td>1015</td>
</tr>
<tr>
<td>VER05/25C</td>
<td>1015</td>
</tr>
<tr>
<td>VER05/25C</td>
<td>1015</td>
</tr>
<tr>
<td>VER05/25C</td>
<td>1015</td>
</tr>
<tr>
<td>VER05/25E</td>
<td>1015</td>
</tr>
<tr>
<td>VER05/25E</td>
<td>1015</td>
</tr>
<tr>
<td>VER05/25I</td>
<td>1015</td>
</tr>
<tr>
<td>VER05/25I</td>
<td>1015</td>
</tr>
<tr>
<td>VER06</td>
<td>1015</td>
</tr>
<tr>
<td>VER06</td>
<td>1015</td>
</tr>
<tr>
<td>VER06</td>
<td>1015</td>
</tr>
<tr>
<td>VER06</td>
<td>1015</td>
</tr>
<tr>
<td>VER06</td>
<td>1015</td>
</tr>
<tr>
<td>VER15/25</td>
<td>1015</td>
</tr>
<tr>
<td>VER15/25</td>
<td>1015</td>
</tr>
<tr>
<td>VER15/25</td>
<td>1015</td>
</tr>
<tr>
<td>VER15/25</td>
<td>1015</td>
</tr>
<tr>
<td>VER15/25</td>
<td>1015</td>
</tr>
<tr>
<td>VER15/25</td>
<td>1015</td>
</tr>
<tr>
<td>VER15/25</td>
<td>1015</td>
</tr>
<tr>
<td>VER34</td>
<td>1015</td>
</tr>
<tr>
<td>VER34</td>
<td>1015</td>
</tr>
<tr>
<td>VER34</td>
<td>1015</td>
</tr>
<tr>
<td>VER34</td>
<td>1015</td>
</tr>
<tr>
<td>VER34</td>
<td>1015</td>
</tr>
<tr>
<td>VER34</td>
<td>1015</td>
</tr>
<tr>
<td>VER34</td>
<td>1015</td>
</tr>
<tr>
<td>VER34</td>
<td>1015</td>
</tr>
<tr>
<td>VER34</td>
<td>1015</td>
</tr>
<tr>
<td>VER59</td>
<td>1015</td>
</tr>
<tr>
<td>VER59</td>
<td>1015</td>
</tr>
<tr>
<td>VER59</td>
<td>1015</td>
</tr>
<tr>
<td>VER59</td>
<td>1015</td>
</tr>
<tr>
<td>VER59</td>
<td>1015</td>
</tr>
<tr>
<td>Model</td>
<td>Package ID</td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
</tr>
<tr>
<td>VER66/14</td>
<td>1015</td>
</tr>
<tr>
<td>VER66/14</td>
<td>1015</td>
</tr>
<tr>
<td>VER66/14</td>
<td>1015</td>
</tr>
<tr>
<td>VER66/14</td>
<td>1015</td>
</tr>
<tr>
<td>VER66/14</td>
<td>1015</td>
</tr>
<tr>
<td>VER66/14</td>
<td>1015</td>
</tr>
<tr>
<td>VER66/14</td>
<td>1015</td>
</tr>
<tr>
<td>VER66/14</td>
<td>1015</td>
</tr>
<tr>
<td>VER66/14</td>
<td>1015</td>
</tr>
<tr>
<td>VER66/14</td>
<td>1015</td>
</tr>
<tr>
<td>VER66/14</td>
<td>1015</td>
</tr>
<tr>
<td>VER66/14</td>
<td>1015</td>
</tr>
<tr>
<td>VER66/14</td>
<td>1015</td>
</tr>
<tr>
<td>VER71 (7000MM)</td>
<td>1015</td>
</tr>
<tr>
<td>VER71 (7000MM)</td>
<td>1015</td>
</tr>
<tr>
<td>VER71 (7000MM)</td>
<td>1015</td>
</tr>
<tr>
<td>VER71 (7000MM)</td>
<td>1015</td>
</tr>
<tr>
<td>VER71 (7000MM)</td>
<td>1015</td>
</tr>
<tr>
<td>VER71 (7000MM)</td>
<td>1015</td>
</tr>
<tr>
<td>VER73</td>
<td>1015</td>
</tr>
<tr>
<td>VER73</td>
<td>1015</td>
</tr>
<tr>
<td>VER97</td>
<td>1015</td>
</tr>
</tbody>
</table>